HIGH TEMPERATURE CREEP

PERFORMANCE OF ALLOY 800H

A thesis

submitted in partial fulfilment

of the requirements for the Degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

IN THE

UNIVERSITY OF CANTERBURY

BY

BENJAMIN GARDINER

University of Canterbury

2014

Preface

This thesis is submitted as a partial requirement for the degree of Doctor of Philosophy in Mechanical Engineering in the University of Canterbury. This research was conducted under the supervision of Professor Milo V. Kral in the Mechanical Engineering Department, University of Canterbury, between March 2009 and March 2014.

Acknowledgments

I would like to acknowledge the following people and organizations that have made this research possible:

- Professor Milo V. Kral for supervision of thesis and for his technical support, knowledgeable critique, optimism, and personal support throughout this research program.
- Co-supervisor Dr Catherine Bishop for her infinite patience, enthusiasm, and technical support.
- Methanex (NZ) Ltd for their financial contributions to this project, and specifically Peter Tait (Methanex) for his assistance and technical advice.
- Members of the Materials Engineering Group (MEG): Emeritus Professor Les Erasmus, Dr. John Smaill, Mike Flaws, Kevin Stobbs, and fellow MEG students.
- Technical staff of the Mechanical Engineering Department, Ken Brown, Julian Philips, Scott Amies, David Read, Gary Cotton and Paul Southward and the numerous administrative staff for their skill, knowledge and kind assistance.
- My family for their support throughout this project.

Abstract

The following document details the investigation into the effect of grain size, grain size distribution, and coherent twins on the creep performance of the stainless steel known as Alloy 800H. Alloy 800H is used for high-temperature piping in industrial applications such as methanol reformers.

The representation and measurement of twinned Alloy 800H microstructures has been achieved through the use of electron backscatter diffraction (EBSD) mapping. The ability to accurately reveal grain boundaries and assess grain boundary types was fundamental to the identification and quantification of coherent twin boundaries, and the measurement of average grain size and grain size distribution. The major advantage of EBSD mapping was the ability to distinguish possible coherent twins using crystal orientation measurement and trace analysis. Grain size measurement from optical micrographs relies on morphological indicators to identify coherent twins. However, it is shown that many of the boundaries observed as straight line morphology on 2D sections did not possess {111} (coherent) interfaces.

3D reconstructions of Alloy 800H revealed the deficiencies in classifying geometry from two-dimensional (2D) sections. Σ3 Crystal volumes were categorized as lamellar or edge structures. Lamellar structures were characterized by the appearance of parallel Σ3 boundary planes while an edge structure contains a single Σ3 interface. Sectioning plane location alters the perception of morphology. For simple twin structures, the tradition 2D classifications of morphology (complete parallel, incomplete parallel and corner Σ3) may all appear on a section plane from a single lamellar structure.

When multiple Σ3 boundaries impinge, the resulting volume can become complex. These crystal volumes have morphologies that are no longer characterised with simple terms such as edge or lamellar. These complex Σ3 volumes were discussed by Reed et al [1] and were described as highly re-entrant shapes that can intersect a sectioning plane many times giving a false impression of multiple separate boundaries.

Alloy 800H processing was performed to investigate the effects of varying strain and temperature combinations on grain size statistics and boundary populations. It was revealed that Σ3 populations may be manipulated by controlling the recrystallization rate. Increasing the nucleation and growth rate of the grains during recrystallization resulted in the formation of microstructures with reduced Σ3 length fractions.

Investigations on post service material showed that Alloy 800H pigtails from methanol producer Methanex have service lives ranging from 3 to 18 years. Because of the variability in service life, a

primary focus of the current study was to investigate creep performance and develop a new criterion for the procurement of Alloy 800H for Methanex pigtails. The current criterion recommends an ASTM grain size of 5 (72µm) or coarser with no consideration given to grain size distribution, grain boundary types, or grain boundary network topology. Results from the investigation showed that this current criterion may produce variations in steady state creep rates of an order of magnitude between ASTM grain size 1 (287µm) and 5 (72µm), and a 2.5 times variation in creep ductility. A revised grain size criterion of ASTM number 2 (203µm) to 3 (143µm) is recommended to provide predictability in creep performance. By maintaining a uniform grain size distribution rupture times of 1.5 to 3.5 times the manufactures data for Alloy 800H are obtainable. Unfortunately no data was obtained regarding the effect of non-uniform grain size distributions on creep performance, a microstructure property often seen to result in early (3 years) pigtail failures.

Contents

List of Figures

List of Tables

Chapter 1 Creep Failure in Alloy 800H Pigtails

1.1 Introduction

Methanex Ltd, the world's largest producer of methanol, utilise steam reformer furnaces in the production of hydrogen and carbon dioxide. The process occurs at temperatures in excess of 850˚C, and as a consequence furnace materials are required to withstand a combination of high temperature and pressure. The current study focuses on the exhaust tubes, called pigtails, constructed of the austenitic stainless steel Alloy 800H.

The deformation mechanism associated with stress at high temperatures is called creep, and understanding the microstructure-creep relationship for different materials is important to Methanex. Understanding how features within microstructures impact creep performance allows Methanex to make informed decisions with respect to material specifications. The ability to predict service life has the potential to reduce unscheduled plant shutdowns, thus reducing cost in replacement, maintenance, and lost production.

Methanex specifies a grain size criterion for certain alloys to help ensure adequate service performance. For Alloy 800H, the default criterion for average grain size is ASTM 5 (72μm) or coarser [2]; several investigations on post service material suggest this may be inadequate. Therefore, Methanex has seen the potential in sponsoring a project with the aim of developing a new criterion based on tighter control of pertinent grain size statistics.

A further focus of this work investigates a feature often neglected in microstructure-property studies: twin boundaries. Twin boundaries are prevalent in many industrially important alloys, although their effect on properties is often ignored, evidenced by their instructed omission from grain size measurement in ASTM E112 [3]. This project will also examine the effect of twin boundaries on the creep performance of Alloy 800H to determine if their inclusion in any future material specification is warranted.

1.2 Methanol Production

Methanol (CH_3OH) is commonly employed as an antifreeze, a solvent, or a fuel in a range of industrial and commercial applications. It also forms the basis for many other products such as recyclable plastics, paints, and explosives. Methanol is predominantly produced from natural gas (CH_4) feedstock, through a process called steam-methane reforming (SMR). Other feedstock, such as coal or biomass, may also be used.

Methanex Ltd. is the world's largest supplier of methanol to the major international markets, including North America, Asia, and Europe [4], and use natural gas as their preferred feedstock. The methanol production process consists of four stages represented schematically in Figure 1.1.

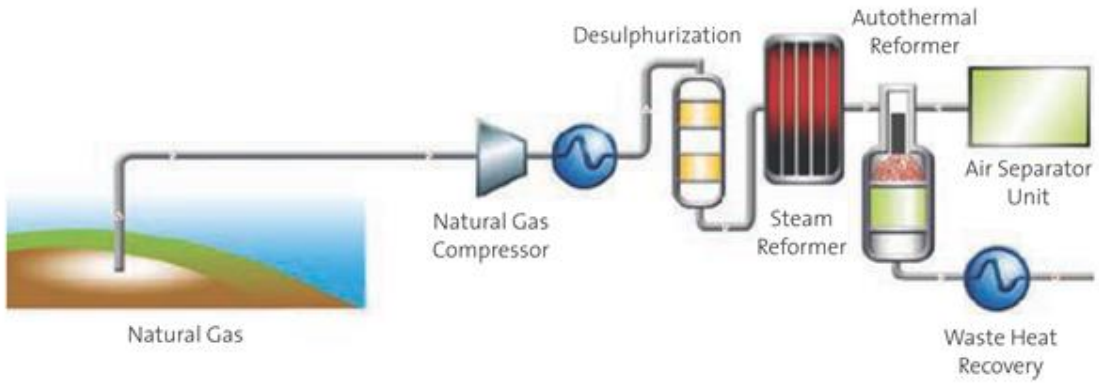
- Desulphurization of natural gas
- Reforming
- Compression and synthesis
- Distillation

A mixture of preheated desulphurized gas and steam enters the steam reformer via an inlet at the top of the furnace. The steam reformer contains many vertical tubes (known as reformer tubes) within a furnace operating at temperatures in excess of 900°C. The steam/natural gas mixture flows through the reformer tubes and across a nickel oxide catalyst producing the steam reforming reactions:

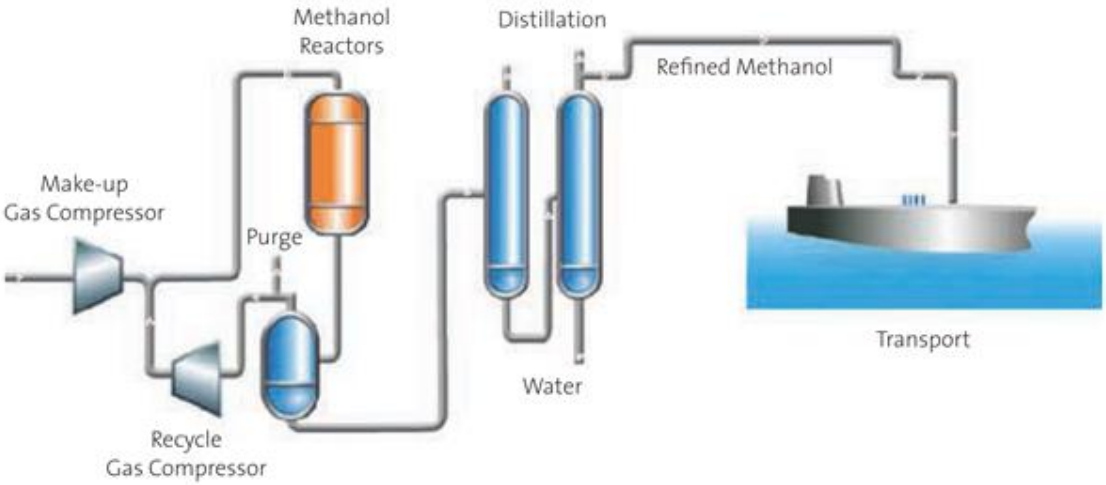
$$CH_4 + H_2O \rightarrow CO + 3H_2$$

$$CO + H_2O \rightarrow CO_2 + H_2$$

The mixture of carbon monoxide (CO), hydrogen (H_2) and carbon dioxide (CO_2) is referred to as synthesis gas, or 'syngas'. The syngas leaves the reformer tubes through welded exhaust tubes called 'pigtails' at an internal pressure and temperature of approximately 1.5MPa and 850°C respectively. The gases are cooled to ambient temperature before entering the next production stage.



(a) Desulphurization and Reforming



(b) Compression and Distillation

Figure 1.1 Stages of methanol production [5]. (a) Desulphurization and Reforming. (b) Compression and Distillation.

1.3 Pigtail Tubes

Due to the high temperatures and moderate pressures associated with the reforming process, significant challenges arise with respect to material performance and component design. Figure 1.2 shows a typical reformer tube/pigtail setup at a Methanex facility, in which the pigtail is located at the bottom of the furnace exhausting the syngas produced within.

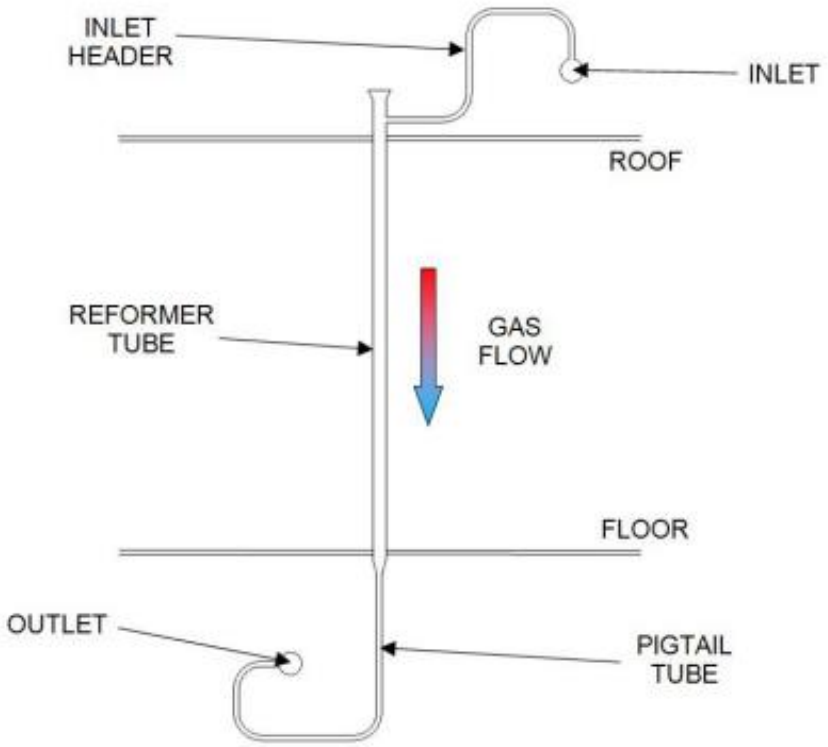


Figure 1.2 Typical reformer tube/pigtail setup. Pigtail tube is shown attached to the reformer tube at the floor of the furnace.

Significant research [6] has been invested to improve the operating life of the reformer tubes, and the success of such work has allowed the effective life of reformer tubes to be extended to approximately 15 years. However, pigtails generally exhibit a useful life of between 10 and 12 years. The differences in design life (and even more concerning some failures after only three years) have resulted in economic loss associated with material cost as well as additional or unscheduled plant shut downs.

1.3.1 Pigtail Geometry and Stress State

The Methanex pigtail tubes measure approximately 42mm outer diameter with a wall thickness of approximately 6mm. The pigtails are produced by either extrusion or cold pilgering. The working cycle is then followed by solution annealing at a high temperature (1150°C) for 30-60 minutes, to meet the grain size specification and dissolve carbon rich precipitates into the matrix.

90° bends are performed by cold bending to obtain the desired configuration. The level of deformation produced by the bending operation is calculated at 10%. The cold work theoretically only exists at the intrados and extrados of the bend, although at the neutral axis of the bend minimum in cold work is expected. After bending, the entire tube is then annealed (30 minutes at 1150°C) in order restore the recrystallized microstructure.

The pigtail tubes experience an internal pressure, and therefore experience a triaxial stress state, illustrated in Figure 1.3. The stresses, σ_t (tangential), σ_x (longitudinal), and σ_r (radial), represent the three directional stresses experienced by the material. The tangential (or hoop) stress is the largest, calculated as 8MPa at the mid wall, and approximately 9.5MPa at the inner surface.

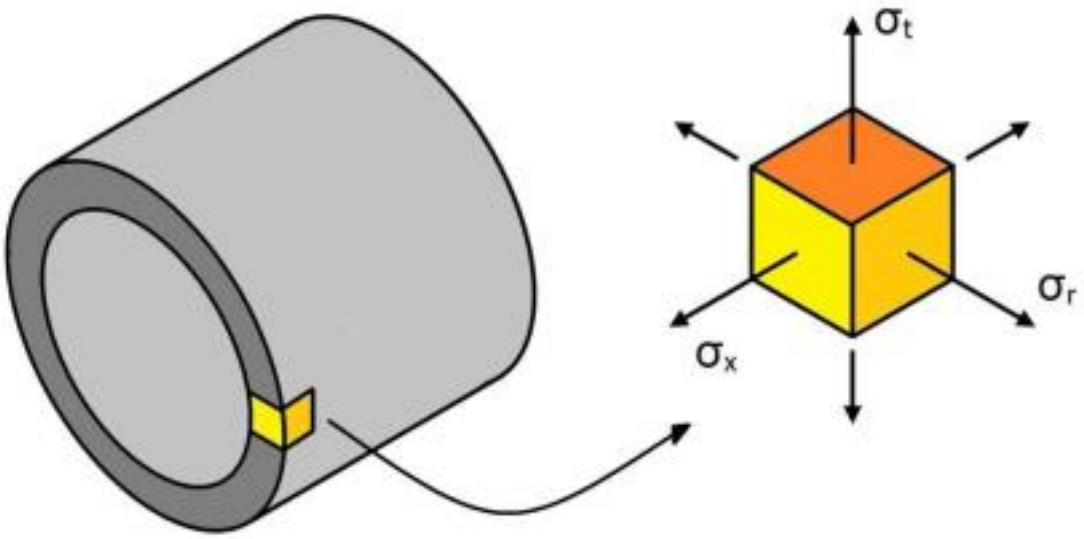


Figure 1.3 Triaxial stress state, σ_t (tangential), σ_x (longitudinal), and σ_r (radial), in a pigtail tube.

1.3.2 Pigtail Grain Structure

To ensure pigtail performance during service, the selected material, Alloy 800H, is purchased with a specified average grain size of ASTM 5 (72µm) or coarser. Unpublished research performed by the University of Canterbury showed that in instances where the average grain size criterion was achieved, the grain size distribution varied significantly between pigtail sources.

Figure 1.4 shows the grain structure for pigtails from four sources: Chile (F4P2), Sumitomo, 1D17, and 25N (Tubacex). Figure 1.5 shows the grain size distributions for the four microstructures from Figure 1.4 plus the distribution for an additional pigtail, 5C12. Compared to 1D17 and Sumitomo, the grain size distributions for Chile (F4P2) and 25N (Tubacex) pigtails indicate an increase in the fraction of grains located at the upper and lower tails. Pigtail microstuctures for Chile (F4P2) and 25N (Tubacex) also showed microstructures with groupings of small grains surrounded by large grains, while the arrangment of grains for 1D17 and Sumitomo appeared uniform.

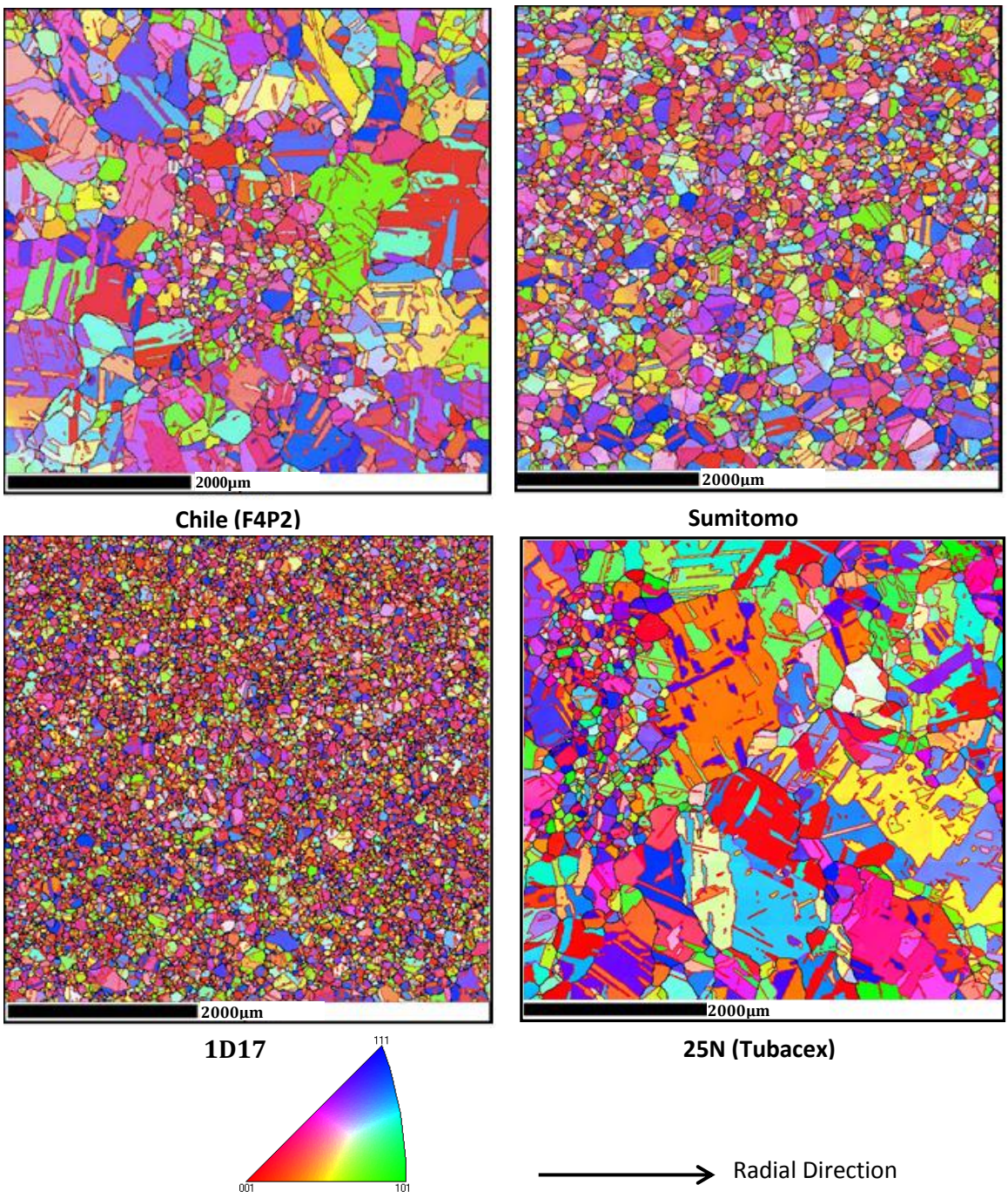


Figure 1.4 Orientation maps representing grain structure of pigtail material from four sources: Chile (F4P2), Sumitomo, 1D17, and 25N (Tubacex). All maps produced from the transverse plane.

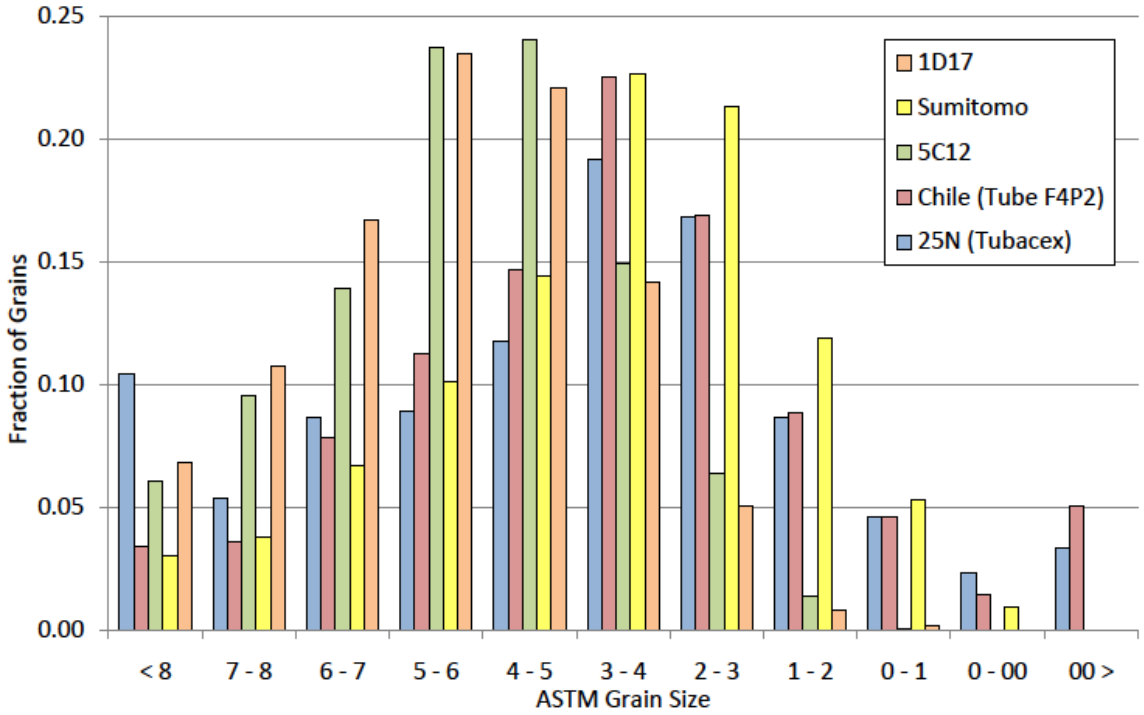


Figure 1.5 Grain size distributions for pigtail material from five sources: Chile (F4P2), Sumitomo, 1D17, 25N (Tubacex), and 5C12.

The service life for the pigtails studied ranged between three years for the Chile pigtail, and 18 years for the Sumitomo pigtail. The results suggest that the grain size distribution and grain arrangement influence creep performance, therefore a criterion solely based average grain size may be insufficient for predicting service performance.

1.4 Material Overview – INCOLOY Alloy 800 Series

The material selected for the pigtail application is a nickel-iron-based stainless steel called INCOLOY Alloy 800H. Alloy 800 (UNS N08800) was first developed in the 1950's as a corrosion and heat resisting alloy with a relatively low nickel content. After years of monitoring alloy performance, the refinement in chemical composition led to improvements in high-temperature strength and resistance to oxidation and carburisation.

The importance of controlling carbon content and grain size to enhance high-temperature properties was recognised early, and with this knowledge Alloy 800H (UNS N08810) was developed. Further research and monitoring resulted in the tightening of carbon limits and additional limits for titanium and aluminium, thus the development of Alloy 800HT (UNS N08811). Table 1.1 summarises the chemical composition of the three grades that make up the INCOLOY Alloy 800 series. Figure 1.6 shows the Fe-Ni-Cr ternary phase diagram showing the composition of Alloy 800H. The diagram indicates that the face centred cubic (fcc) austenite phase is present.

Table 1.1 Compositions, wt.%, for the INCOLOY Alloy 800 series [2].

INCOLOY Alloy	**800**	**800H**	**800HT**
UNS Designation	N08800	N08810	N08811
Nickel	30.0-35.0	30.0-35.0	30.0-35.0
Chromium	19.0-23.0	19.0-23.0	19.0-23.0
Iron	39.5 min	39.5 min	39.5 min
Carbon	0.10 max	0.05-0.10	0.06-0.10
Aluminium	0.15-0.60	0.15-0.60	0.25-0.60
Titanium	0.15-0.60	0.15-0.60	0.25-0.60
Aluminium+Titanium	0.30-1.20	0.30-1.20	0.85-1.20
ASTM Grain Size	Not Specified	5 or Coarser	5 or Coarser

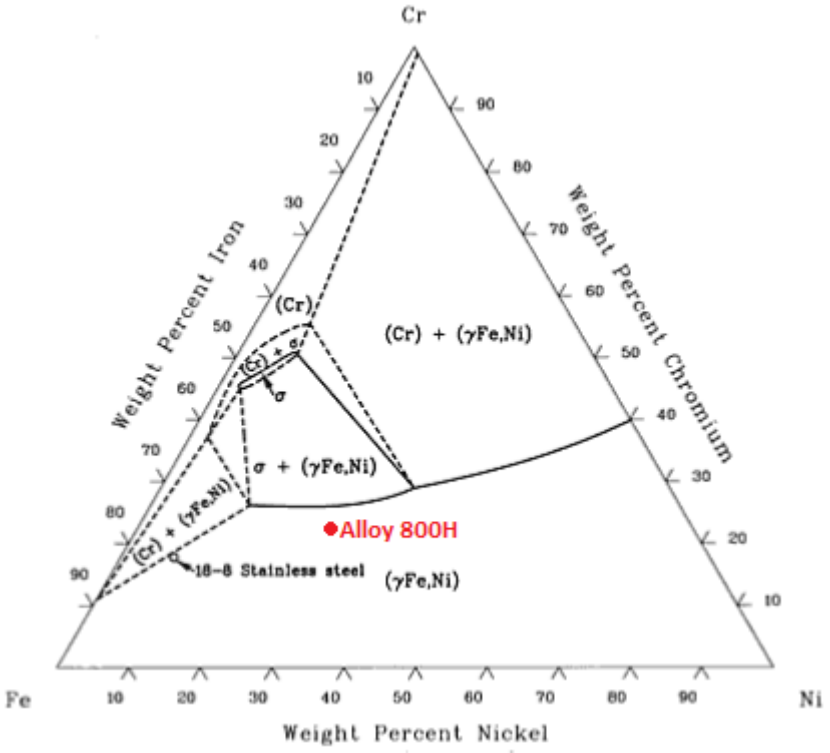


Figure 1.6 Isothermal section at 900°C of the Fe-Ni-Cr ternary phase diagram showing the composition of Alloy 800H in the γFe (austenite) region [7].

The progression in the INCOLOY Alloy 800 series has been the result of ever increasing demands for improved performance at high temperatures, in particular for applications where creep is of concern.

1.4.1 Applications

Since its introduction in the 1950's, the INCOLOY Alloy 800 series has become a workhorse for industrial applications. In 1963 the alloy was approved by the ASME Boiler and Pressure Vessel Committee and design stresses published in Case Code 1325. The alloy series has seen use in furnace components and equipment, sheathing for electrical heating elements, petrochemical furnace cracker tubes, and pigtails and headers.

Alloy 800H is among only four candidate materials that is ASME Code qualified for use in nuclear systems, but only for temperatures up to 760°C and a maximum service time of 300,000 hours. It is proposed that for the Generation IV reactors, the ASME Code extends the allowable service conditions of Alloy 800H to 900°C and 600,000 hours [8].

1.4.2 Alloy 800H Grain Size

Grain size measurement for Alloy 800H pigtails is typically performed using optical microscopy and the intercept method described in ASTM E112 [3]. As instructed by the standard, twin boundaries observed in the optical micrographs are ignored during measurement. The omission of twins during grain size measurement is because the boundary is said to have little or no effect on the mechanical properties of the material. A complete description of twin boundaries, including crystallographic and morphological descriptions, is presented in Section 2.4.

The identification of twins is performed by observing boundary morphology from the optical micrograph, and as a consequence the method is susceptible to operator bias. Observations from EBSD mapping and 3D studies presented in this thesis indicate that 2D morphology observed on optical micrographs suggesting a twin boundary, may be the result from sectioning other boundary types.

The current study implements further investigative techniques to more accurately identity twin boundaries within the Alloy 800H microstructure. The grain size measurements in this study for a occurrences are performed *excluding* twin boundaries unless otherwise stated.

1.4.3 Secondary Phase Formation

Alloy 800H has minor, but important, additions of carbon, aluminium, and titanium. These alloying additions, in combination with the major elements of iron, chromium, and nickel, combine to form a range of second phase particles stable at different operating temperatures [9].

A typical solution annealed microstructure is shown in Figure 1.7. The microstructure reveals a heavily twinned austenitic grain structure with small numbers of titanium carbonitrides (Ti(CN)). These coarse (1 to 10μm) Ti(CN) particles almost always appear yellow in colour indicating they are rich in nitrogen and close to pure titanium nitrides. After aging at temperatures above 600°C these yellow particles are often surrounded by a grey rim (core-rim structure) indicating that existing nitrides have coarsened by growing carbon-rich Ti(CN) on the existing nitrogen rich phase. Due to reasonably low nitrogen content in the alloy (0.05 at.%), very little titanium is consumed by the formation of Ti(CN) allowing for the formation of Ni_3(Ti, Al) (γ') at temperatures between 600°C and 800°C.

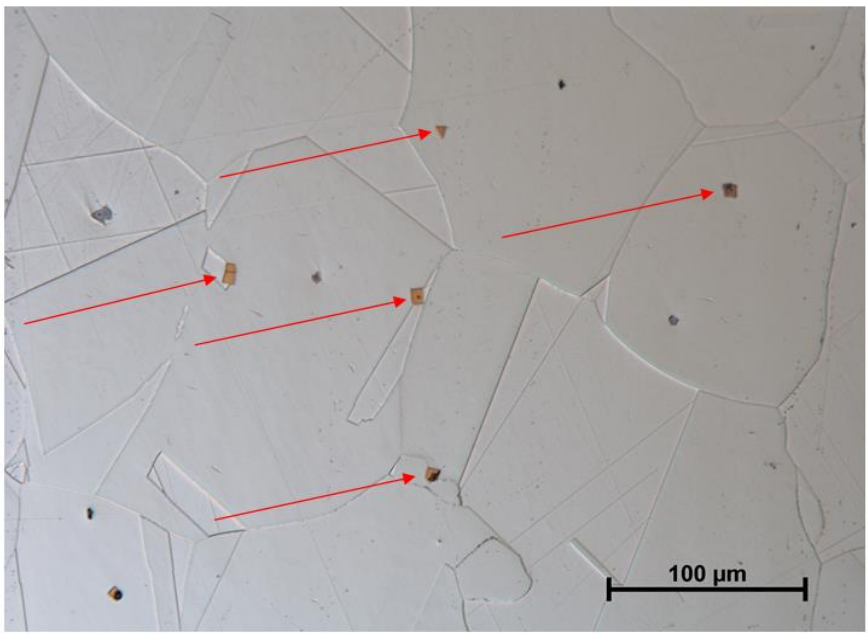


Figure 1.7 Optical micrograph illustrating a typical microstructure of solution annealed Alloy 800H etched with glyceregia. Arrows indicate Ti(CN) particles.

After short aging times above 800°C (i.e. pigtail operating temperatures), the formation of $M_{23}C_6$ is observed preferentially along grain boundaries, as shown in Figure 1.8. Figure 1.9 is the binary phase diagram for Cr and C showing the formation of $Cr_{23}C_6$ between 5.5 and 5.8 wt.% carbon. Because alloy 800H contains a maximum of 0.10 wt.% carbon, and $Cr_{23}C_6$ required 5.5 to 5.8 wt.% carbon to form, $M_{23}C_6$ is typically observed in low volume fractions. The austenite phase dominates the alloy 800H microstructure and has the greatest influence on creep performance.

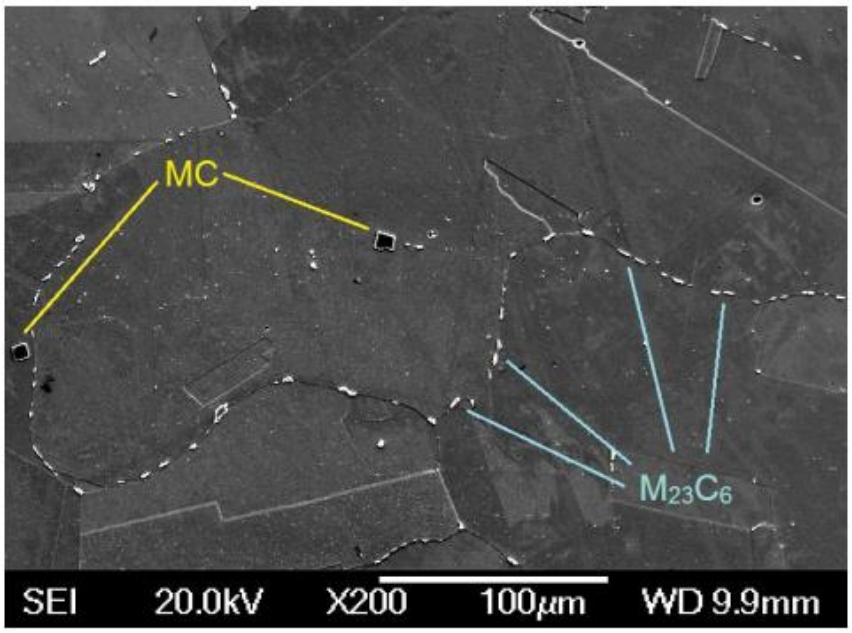


Figure 1.8 Secondary electron image of ex-service alloy 800H, indicating MC and $M_{23}C_6$ carbides.

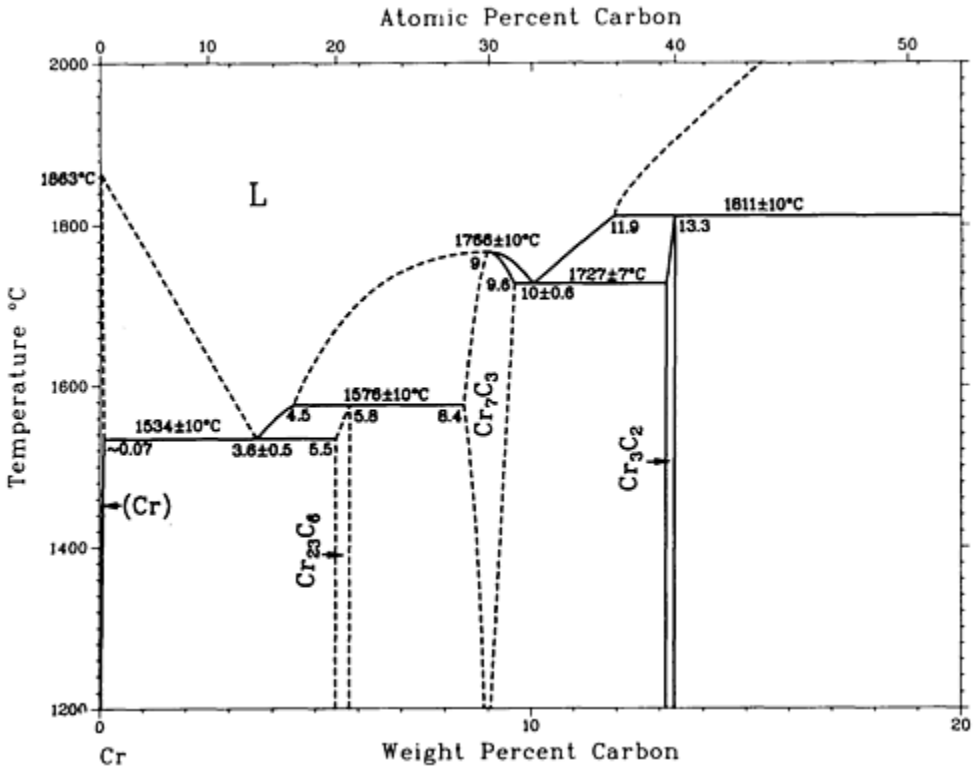


Figure 1.9 C-Cr phase diagram indicating the presence of $Cr_{23}C_6$ at 5.5 to 5.8 wt.% C [10].

1.5 Creep in Alloy 800H Pigtails

1.5.1 Introduction to Creep

Creep is a time dependent, thermally activated process that can occur at stresses below the yield point of the material for the specific operating temperature. Unlike ductile or brittle fracture, creep failure does not occur immediately upon the application of stress, but is 'time-dependent' and strain accumulates over time. Creep rate, $\dot{\varepsilon}$, and total strain at failure, ε_f, are functions of temperature, applied stress, material properties, and the microstructural state, such as grain size.

Creep occurs in three distinct stages: primary, secondary, and tertiary creep; illustrated schematically by a curve representing the typical creep response for a test performed at a constant stress and temperature, Figure 1.10. Primary and tertiary creep stages are known as transient creep since they do not exhibit constant creep rates.

Figure 1.10 Creep response illustrating the three distinct stages of creep: primary, secondary, and tertiary. Steady state creep rate, $\dot{\varepsilon}_{SS}$, and total strain at failure, ε_f, are indicated on the curve.

Primary creep is characterised by high strain rates reducing to a steady-state creep rate indicating the onset of secondary creep. The high strain rates associated with this primary creep stage are explained by the migration of pre-existing dislocations resulting in the formation of sub grain boundaries.

The primary creep stage ends once dislocations are pinned and become immobile. The strain rate drops significantly, and the creep may not continue beyond this point. However, given sufficient stress and thermal energy, the propagation of dislocations and diffusion processes may continue, resulting in secondary creep. Tertiary creep is observed due to the accumulation of creep damage, resulting in an increase in creep rate and the eventual rupture of the sample.

1.5.2 Secondary Creep Mechanisms

Frost and Ashby [11] outlined five primary deformation mechanisms that can act upon a material, with the dominant mechanism being a function of applied stress and temperature.

1. Collapse at the ideal strength – flow when the ideal shear strength is exceeded.
2. Low-temperature plasticity by dislocation glide
    a) Limited by lattice resistance
    b) Limited by discrete obstacles
3. Low-temperature plasticity by twinning
4. Power-law creep by dislocation glide, or glide + climb
    a) Limited by glide processes
    b) Limited by lattice-diffusion controlled climb – *High Temperature Creep*
    c) Limited by dislocation core diffusion controlled climb – *Low Temperature Creep*
5. Diffusion Flow
    a) Limited by lattice diffusion – *Nabarro-Herring Creep*
    b) Limited by grain boundary diffusion – *Coble Creep*

Deformation mechanism maps are a graphical method of summarizing information about the conditions at which each deformation mechanism dominates. These maps are constructed with axes of normalised shear stress, σ_S/μ, and homologous temperature, T/T_M (where σ_s is the applied shear stress, μ is the shear modulus, T is the temperature, and T_m is the melting temperature). The diagram is divided into fields that show the combinations of stress and temperature over which different deformation mechanisms are dominant.

Figure 1.11 is a typical deformation mechanism map constructed 316 stainless steel for an average grain size of 200μm. Power-law creep and diffusional flow are the processes responsible for the secondary creep response. Power-law creep is dominant at high applied stress, while at lower applied stresses diffusional flow is the dominant creep mechanism. The boundaries that separate the fields represent

the stress-temperature combinations at which the deformation mechanisms contribute equally to the creep rate.

Assuming 316 stainless steel provides a suitable approximation for the Alloy 800H system, Figure 1.11 shows the pigtail operating condition would result boundary diffusion being the dominant creep mechanism. With diffusional creep processes dominating, there is an expectation that grain size will have a noticeable influence on creep properties.

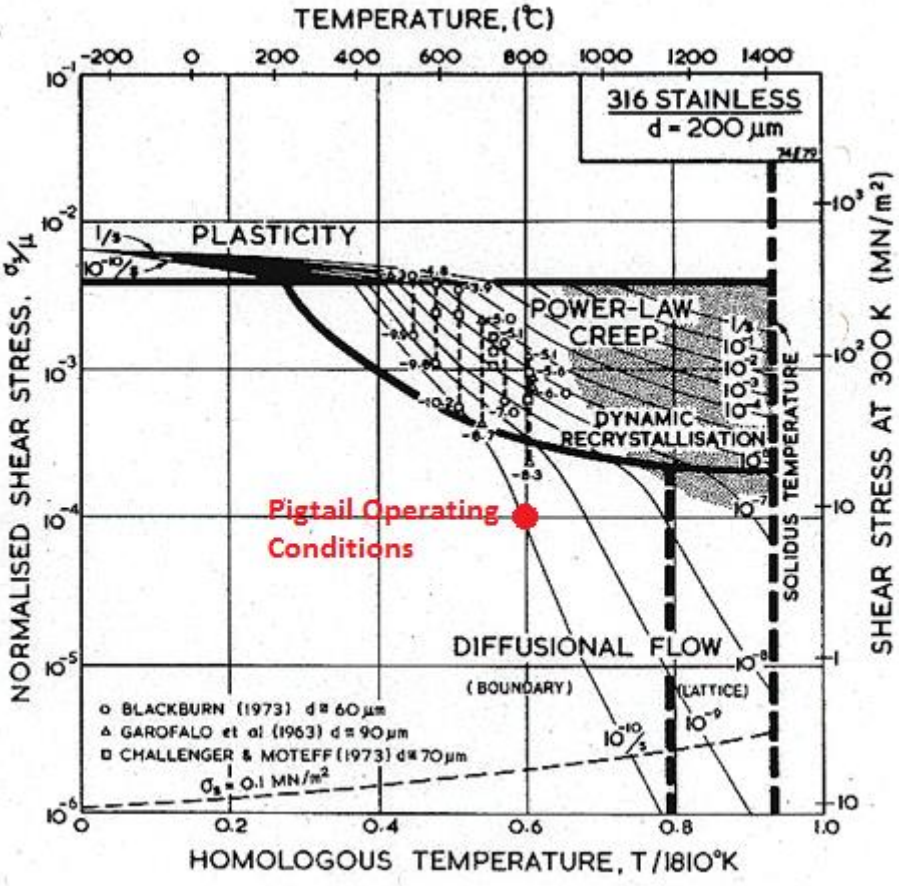


Figure 1.11 A deformation-mechanism map for 316 stainless steel with an average grain size of 200μm [11].

The deformation mechanism associated with power-law creep can be characterised by the activation of recovery mechanisms at elevated temperatures. If thermal energy is sufficient, pinned dislocations become mobile by mechanisms such as dislocation climb. This is why creep becomes prominent at higher temperatures (more thermal energy) and higher applied stress.

At low stresses it is typical to observe diffusional flow processes dominating creep rate. Strain accumulated by diffusional flow is due to the transport of matter from grain boundaries subjected to

compression to those in tension, Figure 1.12. At lower temperatures this process is dominated by the diffusion along grain boundaries (Coble creep [12]), while at higher temperature and larger grain sizes, diffusion though grain interiors (Nabarro-Herring creep [13]) becomes more prevalent.

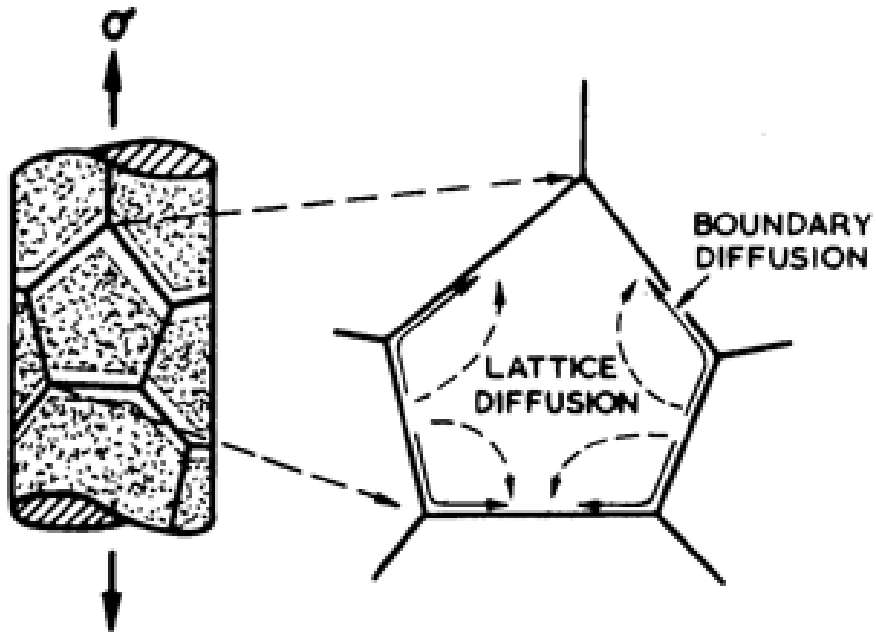


Figure 1.12 Mass diffusion through the grain (Nabarro-Herring) or along boundaries (Coble) [11].

1.5.3 Creep Damage in Pigtails

Investigations show that creep processes are responsible for the majority of damage observed in pigtail tubes. Evidence of creep damage is initially observed by the diametral expansion of the tubes. Microscopic evidence of creep damage exists in the form of voids, which increase in number as creep strain increases. These voids link up along grain boundaries and microcracks begin to form. These microcracks eventually result in through-wall cracking, which constitutes failure of the component. An example of creep damage in the form of voids and microcracks is shown in Figure 1.13, and is accompanied by an example of a typical through wall crack in Figure 1.14. Figure 1.15 shows a 2 mm crack located in the neutral axis of a pigtail that had been in service for less than 4 years.

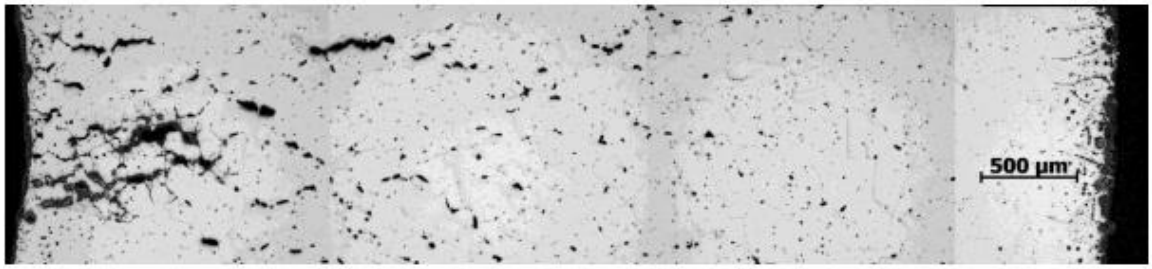


Figure 1.13 Optical micrograph showing creep damage in Alloy 800H pigtail tube showing voids and microcracks [14].

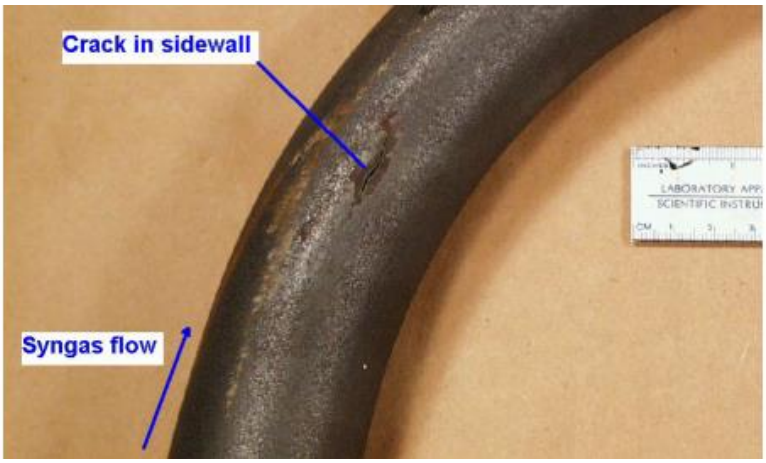


Figure 1.14 Sidewall crack in the neutral axis of an Alloy 800H pigtail [14].

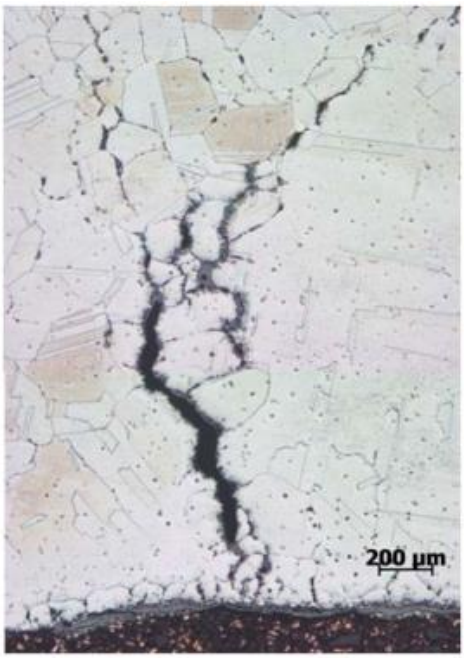


Figure 1.15 Optical micrograph showing a crack at the Alloy 800H pigtail inner wall of the neutral axis of the pipe bend [15].

A number of investigations (e.g. [14]) exist in which pigtails have failed long before their design life. Although a number of factors may have contributed to the failures (including overheating, oxidation, nitriding, and bending stresses due to start-up and shutdown), investigators often suggest non-uniform grain size to be a major issue. In some cases [14], the average grain size has been observed to be two to three ASTM grain sizes larger in the neutral axis compared to the straight sections.

1.5.4 Available Creep Data for Alloy 800H

Creep performance can be displayed by plots of steady-state creep rate versus stress for varying temperatures, as in Figure 1.16. The time to rupture at varying temperatures may also be presented in a similar manner, as in Figure 1.17. The data indicates that an increase in stress or temperature will increase the creep rate and decrease the rupture life.

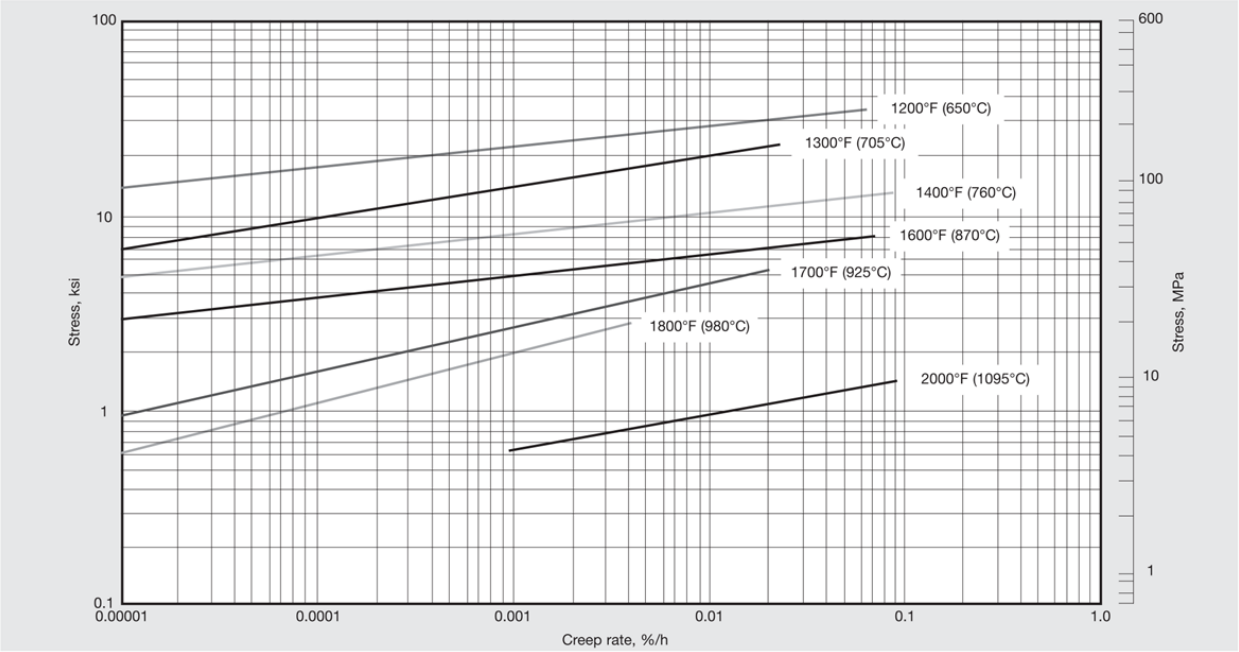


Figure 1.16 Secondary creep rate data for alloy 800H [2].

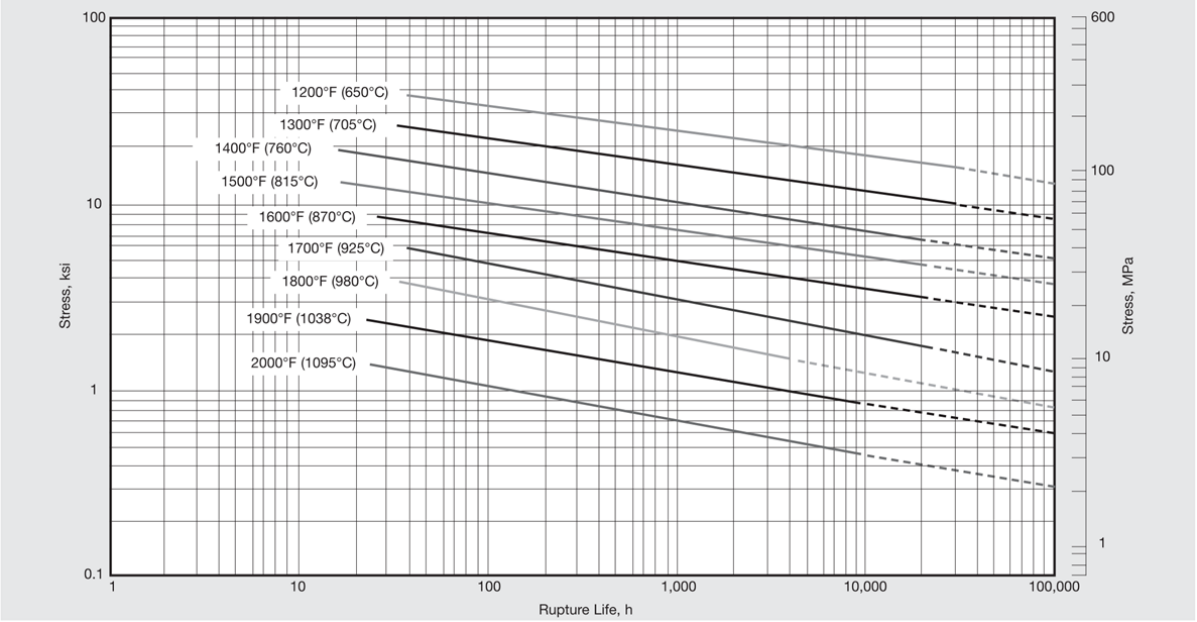


Figure 1.17 Creep rupture life data for alloy 800H [2].

Data presented Figures 1.16 and 1.17 gives no indication of the effect that average grain size, grain size distribution, and twin density may have on the creep properties of Alloy 800H.

1.5.5 Effect of Grain Size on Creep

Grain size has a notable effect on creep rate when diffusional flow mechanisms dominate [16]. The steady-state creep rate, $\dot{\varepsilon}_{ss}$, is expressed through the generalized relationship:

$$\dot{\varepsilon}_{ss} = \frac{AD}{T}\left(\frac{1}{\bar{d}}\right)^{p} {\sigma_s}^{n}$$

Equation 1.1

where A is material dependant constant, D is the diffusion coefficient, T is temperature, $\bar{d}$ is the average grain size, σ_s is the applied stress, and p and n are the exponents for inverse grain size and applied stress, respectively. When dislocation creep processes dominate, typically at high stresses, $p = 0$ and $n \geq 3$. When diffusional flow processes dominate, at low stress and high temperatures, p = 2-3 and n = 1. The value of p = 2 and p = 3 relate to Nabarro-Herring and Coble creep, respectively. Because of this dependence, grain coarsening is often considered an effective way of resisting diffusional creep in polycrystalline materials.

Literature discussing the effect of grain size on the creep properties of Alloy 800H is limited, and the information that exists does not present data above 760°C for the low stress conditions (diffusional creep) [17], or stresses below 100MPa for 900°C (power-law creep) [18]. Figure 1.18 indicates that for a specified stress, the samples with coarser grains will creep at a slower rate at both 650°C and 760°C, a result consistent with Equation 1.1.

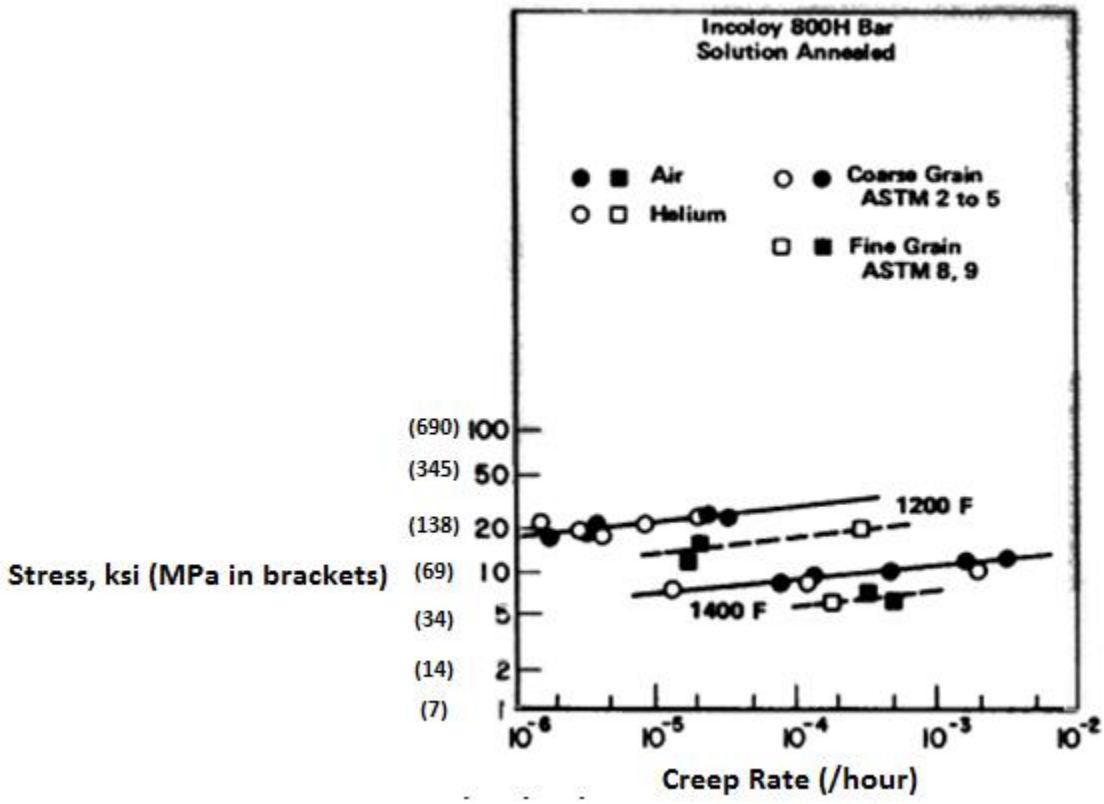


Figure 1.18 Creep rate at 650°C (1200°F) and 760°C (1400°F) for coarse grained (ASTM 2 to 5) and fine grained (ASTM 8-9) 800H samples. Adapted from [17].

While a reduction in the steady state creep rate is typically associated with grain coarsening, this is usually at the detriment of creep ductility. Research suggests that measures such as grain size distributions and largest grains [19, 20] may provide a better indication of creep performance than a single number representing average grain size.

1.5.6 Effect of Grain Size Distribution on Creep

With exception to the investigation briefly discussed in §1.3.2, no information was found discussing the effect of grain size distribution on the creep performance of Alloy 800H. The study (§1.3.2) on the grain structure of the post-service pigtails suggested that a difference in grain size distributions and grain arrangements may significantly influence creep life (15 year difference between samples).

Schneibel et al [21] investigated the extent to which the diffusional creep rate is dependent on the grain size distribution and the arrangement of grains. A simple model employing grains of two different sizes showed that, depending on the exact arrangement of the different sized grains, internal stresses may be 3.5 times higher than the applied stress. Larger stresses were typically seen in arrangements where a large grain is surrounded by several smaller grains.

Both Onaka et al [22] and Kim et al [20] employed energy balance methods to analyse the effect of grain size distributions on the steady state creep rate due to diffusional creep. Onaka concluded that modelling creep rates without considering grain size distribution may overestimate the steady state creep rate by as much as two times for diffusional creep. Kim also concluded that the inclusion of the grain size distribution into diffusional creep models would produce steady state creep rates less than those predicted when only the average grain size was considered.

Using the diffusion-based rate equations [11], the effects of average grain size and grain size distribution on the steady state creep rate were modelled for Alloy 800H (Appendix A). Figure 1.19 shows the results for a stress of 13.5MPa and temperature of 980°C, for a range of average grain sizes. The red curve represents the steady state creep rate where the grain size distribution is neglected. The green curves represent the steady state creep rates for three different values of D: 0.6, 0.8, and 1.0, where D is the standard deviation of the variable's natural logarithm describing the width of the grain size distribution and is given by:

$$D = \sqrt{\ln\left(1 + \frac{S^2}{\bar{d}^2}\right)}$$

Equation 1.2

where $\bar{d}$ and S are the arithmetic mean and standard deviation of the non-logarithmized grain size measurements respectively. S is given by:

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \bar{d})^2}$$

Equation 1.3

The results show that by incorporating the grain size distribution into the model, creep rates may be up to three times slower than predicted based on average grain size alone. This finding is consistent with previous work [20, 22].

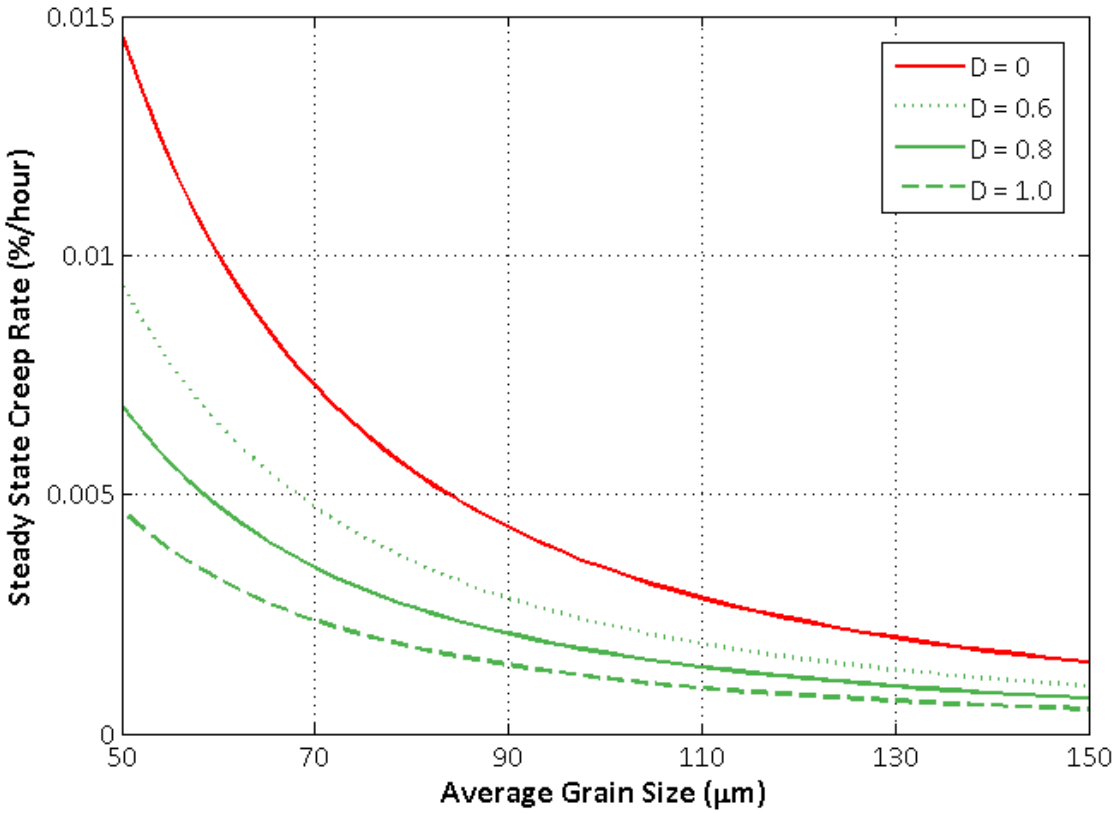


Figure 1.19 Steady state creep rates for Alloy 800H for a stress of 13.5MPa at 980°C. The red curve ($D = 0$) represents the predicted creep rates for average grain size only, and the green curves represent the predicted creep rates for $D = 0.6$-1.0.

1.5.7 The Effect of Twin Boundaries on Creep

Limited studies have been published analysing the effect of twin boundaries on creep performance. Most studies (e.g. [23-25]) tend to analyse the resistance of twins to the formation of cracks and cavitation. Otto et al [26] recognised that Σ3 boundaries (of which twins are a subset) and Σ9 boundaries tend to show greater resistance to creep cavitation in copper and copper alloys.

Boehlert [27] examined the creep rate of Alloy 690 at temperatures between 650°C and 690°C, and stresses between 75MPa and 172MPa. The samples with fewer twins and smaller grain size exhibited an increased creep rate. The combination of fewer twins coupled with a smaller grain size makes it impossible to conclude whether or not the twin density had an effect on creep rate.

Cui et al [28] investigated the effects of alloying and heat treatment on the creep performance of Ni-Co based super alloys. Higher concentrations of cobalt decreased stacking fault energy and increased the twinning frequency. While the increase in twin density improved creep performance, the effects of alloying on creep rate were not discussed.

A grain boundary engineering (GBE) study of Alloy 800H performed by Drabble [15] showed that minimum creep rates could be reduced by up to 30% compared with material in the as-received condition. While GBE reduced the diffusivity of the grain boundary network by breaking up the high angle grain boundary network, an increase in average grain size from 120µm to 190µm was also noted to have possibly been a contributing factor.

1.6 Research Outline

1.6.1 Research Hypotheses

After substantial background reading and discussions with sponsor Methanex a project scope was developed. The project scope combined the aspirations of Methanex to better understand the effect of grain size on the creep performance of Alloy 800H, and an academic objective of developing a greater understanding of the structure and formation of twined microstructures.

The three main research objectives are:

1. **Investigate the morphology of Alloy 800H microstructure.**
2. **Examine how processing conditions affect the formation of twin boundaries.**
3. **Investigate the relationship between the austenitic grain structure and the creep response of Alloy 800H for Methanex pigtails.**

To assist in developing a new Alloy 800H criterion for Methanex, a greater understanding of the Alloy 800H microstructure is required. By performing serial sectioning and 3D reconstructions on Alloy 800H we advance the understanding of grain structure and twin boundary morphology. The insights generated by the 3D reconstructions assist in the development of methodologies to measure grain size and grain boundary character.

The 3D reconstructions are predicted to reveal the complexity of intersecting twin interfaces and the non-trivial morphologies that result. The study will also highlight that 2D boundary morphologies observed on a section plane are not a perfect indicator of interface plane inclination. The straight line morphology of boundaries often associated with coherent twin boundaries ({111} planar normal) may possibly be asymmetrical tilt boundaries around a <110> direction. While faceting along twin boundaries is often observed in 2D, it is predicted that the degree of faceting may be more significant when boundary interfaces are observed in 3D. The 3D study will reveal that boundary faceting is an important part in interface energy minimization by producing geometrically favourable interface planar normals and triple junctions.

A study examining the effect of cold work and annealing temperature on twin boundary populations in Alloy 800H is performed. The results from the study, in conjunction with 3D analysis, assist in developing an understanding of how and why twin boundaries form. Previous Alloy 800H processing has indicated that low strains (6%) followed by annealing at moderate temperatures (<1250°C) was able to produce large populations of twin boundaries resulting in the discontinuity of the random high angle grain

boundary network. It is hypothesised that in the current study large cold-work strains (>50%) combined with high annealing temperatures (>1300°C) will produce Alloy 800H microstructures will measurably fewer twin boundaries.

Alloy 800H is ordered with a specified average grain size of ASTM 5 or coarser to ensure resistance against creep and a predictable service life. However, investigations on post service material showed that pigtails meeting this specification had service lives ranging from three to 18 years. In addition to re-evaluating the suitability of the current grain size criterion, the effect of grain size distribution and twin boundaries on creep performance will also be assessed to determine whether their inclusion in future material specification is necessary.

The high temperature-low stress creep conditions applicable to Methanex operating conditions indicates that an obvious correlation between average grain size and creep rate will be established in the current study. Analysis of post service pigtail samples has given a strong indication that creep performance is also related to grain size distribution. It is hypothesised that samples with grain size distributions skewed toward the lower tail (higher frequency of small grains) will result in an increase in creep rate. Samples with a higher frequency of larger grains will initially indicate superior performance with a decrease in creep rate, but overall performance will be determined by ductility at failure.

1.6.2 Research Overview

A progress chart of the research program is outlined in Figure 1.20.

Electron Backscatter Diffraction (EBSD) was used extensively throughout this course of study. Therefore, the first task was to investigate the techniques and methods to acquire EBSD data. This area of study not only examined the metallographic preparation techniques required to produce quality EBSD data, but also developed the methods used analyse the raw data and provide results pertaining to boundary types and grain size. The methods underwent several refinements over the course of study.

Thermo-mechanical processing of Alloy 800H was intended to provide a greater understanding of how varying amounts of strain and temperature can affect the formation of coherent twin boundaries. Although a large amount of literature already existed on this topic, it was often found to be inconsistent or lacking a rigorous methodology with respect to identifying or quantifying twin populations.

To fully appreciate and understand the two-dimensional representations of twin boundaries observed on planar sections, knowledge of their three-dimensional morphology was required. Without this knowledge, it is difficult to accurately quantify the grain size and twin boundary population within the microstructure. For this reason, a serial sectioning study was conducted resulting in the refinement of the initial methodologies.

Finally, creep testing was performed on Alloy 800H samples prepared with varying grain sizes, grain size distributions, and twin frequencies in an attempt to better understand the structure-property relationship.

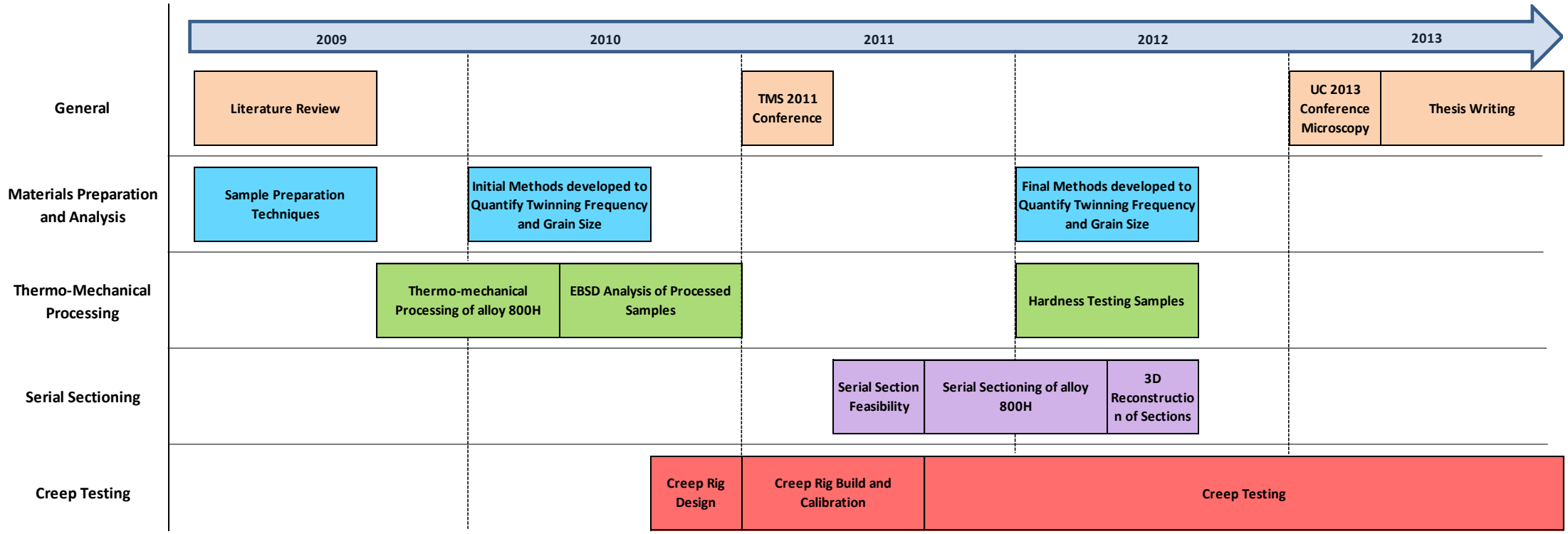


Figure 1.20 Progress chart of overall research program.

1.6.3 Research Achievements

- Milo V. Kral, Daniel J.F. Drabble, Benjamin R. Gardiner and Peter C. Tait, *"Implications of EBSD-based Grain Size Measurement on Structure-Property Correlations",* Praktiche Metallographie (2009), 46, 9, 469-482
- Benjamin R. Gardiner and Milo V. Kral, *"The Effect of Grain Size and Annealing Twin Density on the Creep Properties of Alloy 800H",* presented at TMS Annual Meeting, February 27 - March 3, 2011, San Diego, CA, USA
- Milo V. Kral and Benjamin R. Gardiner, *"Morphology and Crystallography of Annealing Twins in Austenite",* 1st International Conference on 3D Materials Science, July 8-12,2012, Seven Springs, PA, USA
- Benjamin R. Gardiner and Milo V. Kral, *"Morphology and Crystallography of Annealing Twins in Austenite",* 26th New Zealand Conference on Microscopy, February 13-15, 2013, Christchurch, NZ
    - Awarded the Keith Williamson medal for excellence in research involving microscopy
- Milo, V, Kral and Benjamin R. Gardiner, *"Prediction of Creep Life of Alloy 800H using EBSD Grain Size Measurement Methods",* IMS Quantitative Metallography Conference and Exposition, April 3-5, 2013, San Antonio, TX, USA
- Milo. V. Kral and Benjamin R. Gardiner, *"Prediction of Creep Life of Alloy 800H using EBSD Grain Size Measurement Methods",* The 8th Pacific Rim International Conference on Advanced Materials and Processing, August 4-9, Waikoloa, HI, USA

1.7 Thesis Outline

This thesis is divided into seven additional chapters, each briefly summarised as follows:

Chapter 2 provides an overview of the recrystallization process, in particular how it pertains to the formation of grain structure and boundary types. An overview of grain boundaries is provided, including an introduction to Electron Backscatter Diffraction (EBSD) techniques and the coincident site lattice (CSL) model used to describe grain boundary geometry and identify grain boundary types. A review of twin boundaries is presented, including structure, formation theories, and methods used to identify twins within a microstructure.

Chapter 3 provides details on the techniques employed to obtain quantitative measures of Alloy 800H microstructures. In general, the metallographic preparation procedures for austenitic stainless steels are well developed and understood. Therefore, this chapter focuses primarily on EBSD mapping procedures and the algorithm developed to extract the necessary information from the raw EBSD data.

Chapter 4 presents methodologies, results, and discussion related to the serial sectioning and three-dimensional reconstruction of a typical Alloy 800H microstructure. This chapter provides information related to 3D morphologies and the crystallographic descriptions of twin interfaces. The information provided by the three-dimensional data set provided a means to validate the procedures discussed in Chapter 3 for measuring grain size and identifying twin boundaries.

Chapter 5 analyses the effect of varying strains and temperatures to alter the recrystallization and grain growth dynamics in an attempt to control grain boundary character and grain size distribution.

Chapter 6 describes the creep testing apparatus and methodology, and presents the results from the creep testing performed on Alloy 800H. Emphasis is on the selection of the appropriate test conditions and the effect of grain sizes and grain size distributions on the creep performance of Alloy 800H.

Chapter 7 provides a summary of the results and conclusions for the entire body of work in relation to the research objectives listed above.

Chapter 8 discusses potential future research topics which stem from the current study.

Chapter 2 Background Literature

2.1 Introduction

Essential to the present study is an understanding of how Alloy 800H microstructures develop grain size distributions and twin populations through recrystallization. Recrystallization is the formation of new grains from a deformed microstructure. Thermal-mechanical processes influence the nucleation and growth rate of grains during recrystallization, and thus impact the final grain size, grain size distribution, and density of boundary types.

Also discussed is the identification of grain boundary types through the measurement of boundary geometry using Electron Backscatter Diffraction (EBSD). The geometry of a grain boundary is defined by the misorientation between the adjacent crystal lattices and the inclination of the interface plane. The coincident site lattice (CSL) model is a method of classifying certain boundary misorientations which have the potential to exhibit low energy and unique properties.

The Σ3 boundary is an example of a boundary with a misorientation described by the CSL model. The coherent twin, a subset of Σ3, is characterised by its low energy. The coherent twin is not only defined by the misorientation across the boundary, but also its boundary plane, the geometry of which cannot be measured by EBSD on a single surface. 3D reconstruction of serial sections provides a way to analyse the 3D morphology of twin boundaries and measure the boundary plane orientation.

Finally, the theories describing the mechanisms by which twin boundaries forms are reviewed. A focus is on understanding how deformation and annealing temperature effect the formation of twin boundaries. The formation of twin boundaries is shown to play a role in the recrystallization process.

2.2 Recrystallization

Recrystallization is the nucleation and growth of new grains from a deformed microstructure. Growth is achieved by the migration of high angle grain boundaries driven by stored energy. Recrystallization restores the ductility of the material hardened by low temperature deformation (deformation occurring below 50% of the absolute melting temperature), and controls the grain structure of the final product.

2.2.1 Deformation and Recovery

The deformed microstructure influences the nucleation and growth rates of new grains during recrystallization. The deformed microstructure is characterised by the division of existing grains by the formation of dislocation boundaries [29]. A deformed microstructure consists of blocks of dislocation cells surrounded by long flat dislocation boundaries. Figure 2.1(a) shows that for low to medium strains (<50%) the dislocation cells appear equiaxed. The dislocation boundaries surrounding the blocks of cells are called single walled dense dislocation walls (DDWs) and double walled microbands (MBs). At higher strains, the cell blocks are flattened between lamellar dislocation boundaries (LBs), as shown in Figure 2.1(b). The dislocation cell boundaries tend to have low misorientation angles, while the DDWs, MBs, and LBs tend to have boundaries with higher misorientations. As strain increases, the misorientation of the dislocation boundaries increases.

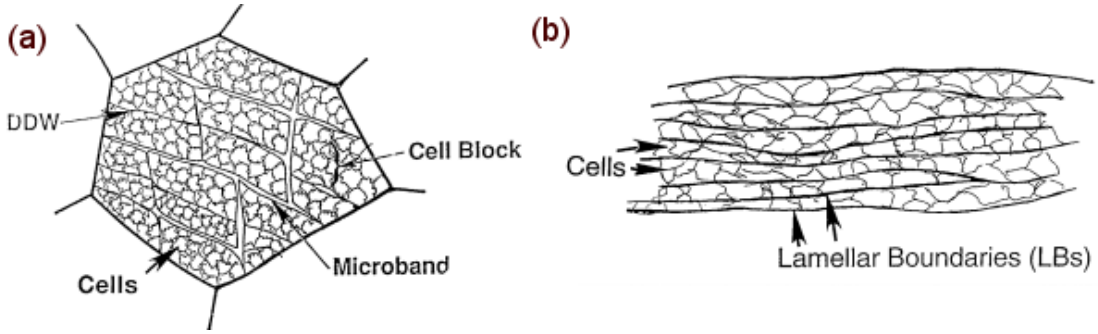


Figure 2.1 Schematic representations of grain subdivision for small (a) and large (b) strains [29].

Recovery is the rearrangement of dislocations to form low angle grain boundaries (LGBs). Strain free regions form due to the rearrangement of dislocation. These strain free regions act as ideal nucleation sites for the formation of new grains during recrystallization. An important feature of the recovery process is that the structural changes occurring do not involve the migration of high angle grain boundaries (HGBs).

2.2.2 Recrystallization

Recrystallization is a thermally activated process in which new grains nucleate and grow at the expense of the deformed matrix. The driving force for recrystallization is the reduction in free energy, achieved by the migration of HGBs through the deformed microstructure. Recrystallization can be described as a two-step process: the nucleation of new grains and the growth of the nucleated grains.

The Johnson-Mehl-Avrami-Kolmogorov (JMAK) model [30, 31] describes the kinetics of recrystallization in terms of nucleation rate, $\dot{N}$ (the number new grains formed per unit volume) and the growth rate of the new grains, G.

Initially ignoring boundary impingements during growth, the number of recrystallized nuclei formed is given by:

$$dn_{ex} = \dot{N}dt$$

Equation 2.1

where dn_{ex} is the *extended* number of nuclei formed over time, dt. Extended refers to the assumption that nuclei can overlap with each other as they grow, i.e., impingement is ignored. The extended volume fraction, $f_{ex}(t)$, is therefore given by:

$$f_{ex}(t) = \int_0^t V\dot{N}dt$$

Equation 2.2

Assuming nuclei are spheres, the volume, V, is given by:

$$V = \left(\frac{4\pi}{3}\right)[G(t - t_0)]^3$$

Equation 2.3

Combining Equation 2.2 and 2.3 with the assumption $t_0 \ll t$ gives:

$$f_{ex}(t) = \int_0^t \left(\frac{4\pi}{3}\right) G^3 t^3 \dot{N}dt = \left(\frac{\pi}{3}\right) G^3 t^4 \dot{N}$$

Equation 2.4

The actual number of nuclei forming during recrystallization is less than dn_{ex} because nuclei cannot form in those parts of the specimen which have already been recrystallized so therefore boundary impingement restricts growth. The relationship between the change in the true recrystallization fraction, $df(t)$, and the change in the extended fraction, $df_{ex}(t)$, is given by:

$$df_{ex}(t) = \frac{df(t)}{1 - f(t)}$$

Equation 2.5

where $f(t)$ is the true fraction of recrystallized material. Solving Equation 2.5 for $f(t)$, and combining with Equation 2.4, gives:

$$f(t) = 1 - \exp\left(\frac{-\pi}{3}\dot{N}G^3t^4\right)$$

Equation 2.6

Assuming recrystallization is complete at $f(t)$ = 0.95, Equation 2.6 can be solved for time:

$$t = \left(\frac{2.86}{\dot{N}G^3}\right)^{\frac{1}{4}}$$

Equation 2.7

The number of nuclei formed per unit volume is given by:

$$\dot{N}t = 1.3\left(\frac{\dot{N}}{G}\right)^{\frac{3}{4}}$$

Equation 2.8

If d is the average diameter of the recrystallized grain, the number per unit volume will be approximately equal to $1/d^3$, so that:

$$d = 0.92\left(\frac{G}{\dot{N}}\right)^{\frac{1}{4}}$$

Equation 2.9

From Equation 2.9 it is apparent that the ratio $^{G}/_{\dot{N}}$ will influence the recrystallized grain size. Coarse recrystallized grain size can be obtained by maintaining a high $^{G}/_{\dot{N}}$, ratio, that is, low nucleation rate and fast growth rate.

Figure 2.2 represents the change in recrystallization fraction over time as described by Equation 2.6 for three temperatures: $T_1 > T_2 > T_3$. An increase in temperature will result in faster nucleation and growth rates, thus reducing recrystallization time. The temperature at which a given sample will completely recrystallize in a specific time (usually one hour) is called the recrystallization temperature, i.e., the recrystallization temperature in Figure 2.2 is T_1. For practical reasons, $f(t)$ = 0.95 is considered to be the completion of recrystallization. Temperature has a greater influence on the growth rate then the nucleation rate, therefore, an increase in temperature would result in an increase in the ratio $^{G}/_{\dot{N}}$, resulting in a coarser recrystallized grain size.

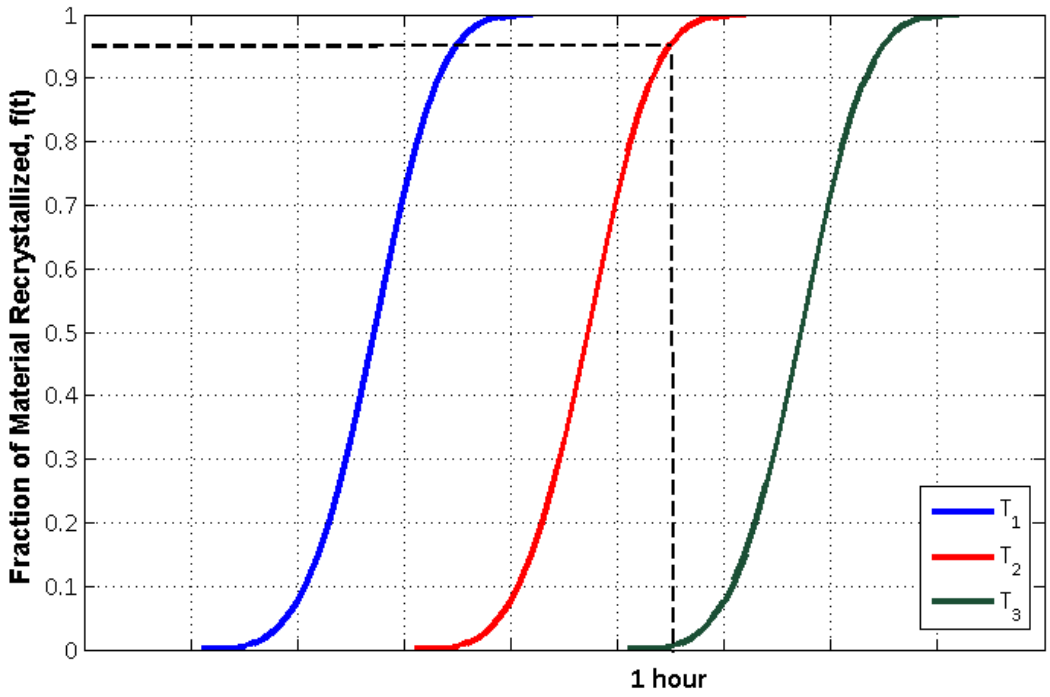


Figure 2.2 Recrystallization fraction, $f(t)$, for three annealing temperatures: $T_1 > T_2 > T_3$.

A minimum amount of deformation is required to initiate recrystallization. The deformation must be high enough to provide sites for the formation of the nuclei, and provide a driving force for their growth. An increase in deformation provides a greater number of dislocation boundaries, ideal locations for the nucleation of grains. While the additional stored energy, due to the larger deformation, will accelerate growth rate, a greater increase is observed in the nucleation rate, resulting in a finer recrystallized grain size.

Figure 2.3 shows the relationship between initial grain size (grain size prior to deformation) and recrystallized grain size in brass for three deformations: 20.5%, 40.4%, and 61.8%. Figure 2.3 shows that for a constant initial grain size, the recrystallized grain size decreases with increasing deformation. Finer initial grain sizes also provide a greater number of high angle grain boundaries, sites from which new grains can nucleate. Therefore, an increased nucleation rate is observed for specimens with finer initial grain size, resulting in a finer recrystallized grain size.

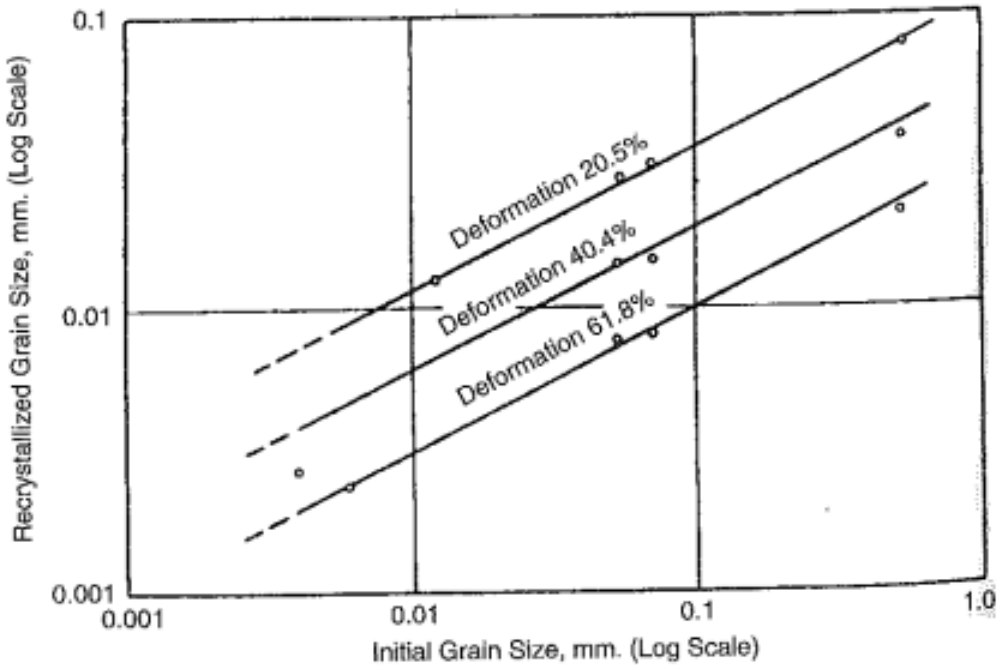


Figure 2.3 Relationship between recrystallized grain size and initial grain size in 70-30 brass for different values of prior deformation [32].

2.2.3 Grain Coarsening

Annealing after recrystallization will increase the average grain size. This process, called grain coarsening, is driven by the surface energy in grain boundaries. Burke and Turnbull [33] derived the parabolic equation for the kinetics of grain growth.

$$\bar{d}^2 - \overline{d_0}^2 = kt\, exp\left(-\frac{Q}{RT}\right)$$

Equation 2.10

where $\bar{d}$ is the average grain size after time t at temperature T. $\overline{d_0}$ is the initial grain size, Q is the activation energy, and k is the growth constant. The model was proposed for the case of normal grain

growth in a pure homogeneous material. Often experimental results do not obey the parabolic equation. For this reason, Equation 2.10 is typically used in the form

$$\overline{d}^n - \overline{d_0}^n = k_0 t \, exp\left(-\frac{Q}{RT}\right)$$

Equation 2.11

where n is the grain growth exponent and k_0 is a fitting constant. Equation 2.11 is referred to as the power law for grain growth.

Anderson [34] summarised the results of grain growth experiments for pure metals (Al, Fe, Pb, and Sn) with grain growth exponents ranging from 2 to 4. The deviation from parabolic growth kinetics ($n = 2$) is because the original assumption by Burke and Turnbull that the only forces which influence boundary motion are due to surface curvature is typically not the case. Atkinson [35] suggests that the mechanisms which influence the value of *n* include the presence of solute atoms as well as secondary phases pinning and slowing the migration of grain boundaries.

2.2.4 Grain Size Distribution

Histograms are used to analyse the grain size distribution. To present grain size data as a histogram, discrete size groups (class intervals) are employed [36]. Grains of similar size are grouped together in a single class interval represented by a mean class grain size. The average two-dimensional grain size $\bar{d}$ for grains grouped into class intervals can be given by Equation 2.12, where $(N_A)_j$ is the number of grains in class interval j, with mean diameter d_j, and N_A is the total number of grains.

$$\bar{d} = \frac{1}{N_A}\left((N_A)_1 d_1 + (N_A)_2 d_2 + \cdots + (N_A)_j d_j + \cdots + (N_A)_{max} d_{max}\right)$$

Equation 2.12

Figure 2.4 is an example of a grain size distribution histogram for Alloy 800H for which a class interval of 20µm has been used. For example, the first bar of the histogram represents the proportion of grains that exist in the distribution with grain size between 0µm and 20µm.

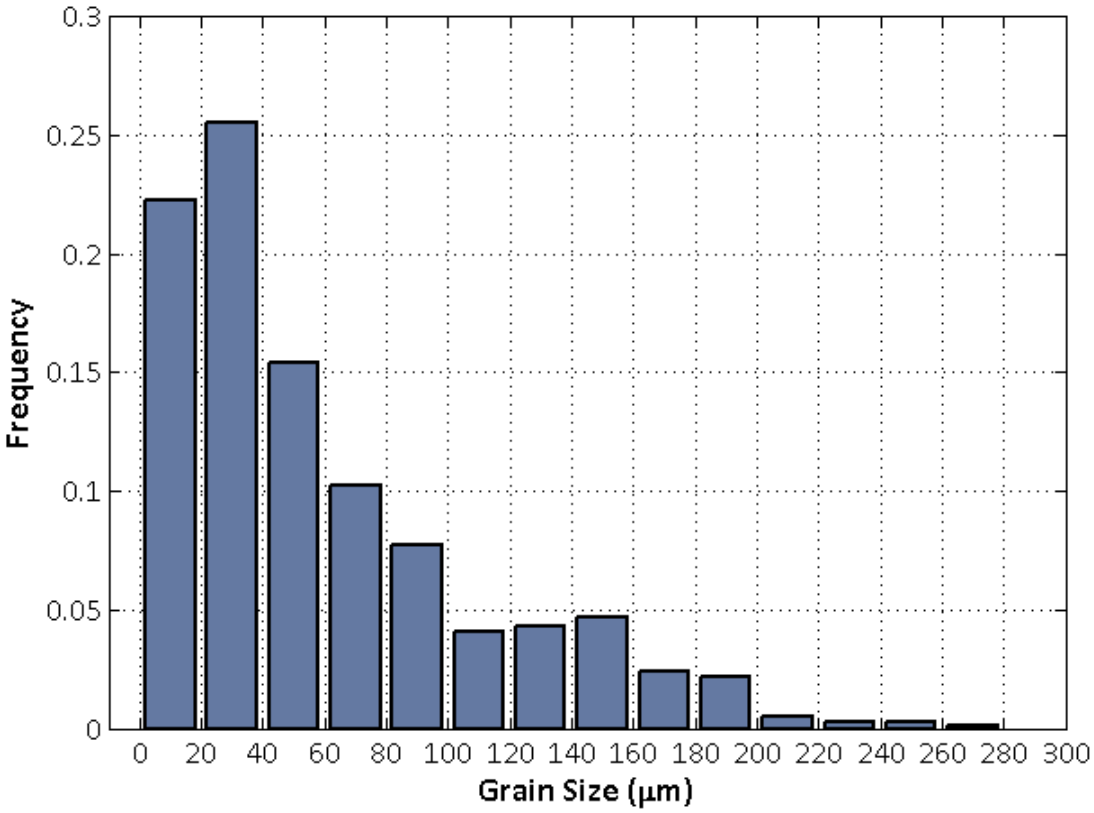


Figure 2.4 Grain size distribution histogram for As-Received Alloy 800H grouped in class intervals of 20μm.

The grain size distribution of polycrystals formed by normal grain growth can be represented by the lognormal distribution [36-40]. The lognormal distribution is a positively skewed unimodal distribution, Figure 2.4, which when plotted on a logarithmic scale has a Gaussian form, Figure 2.5.

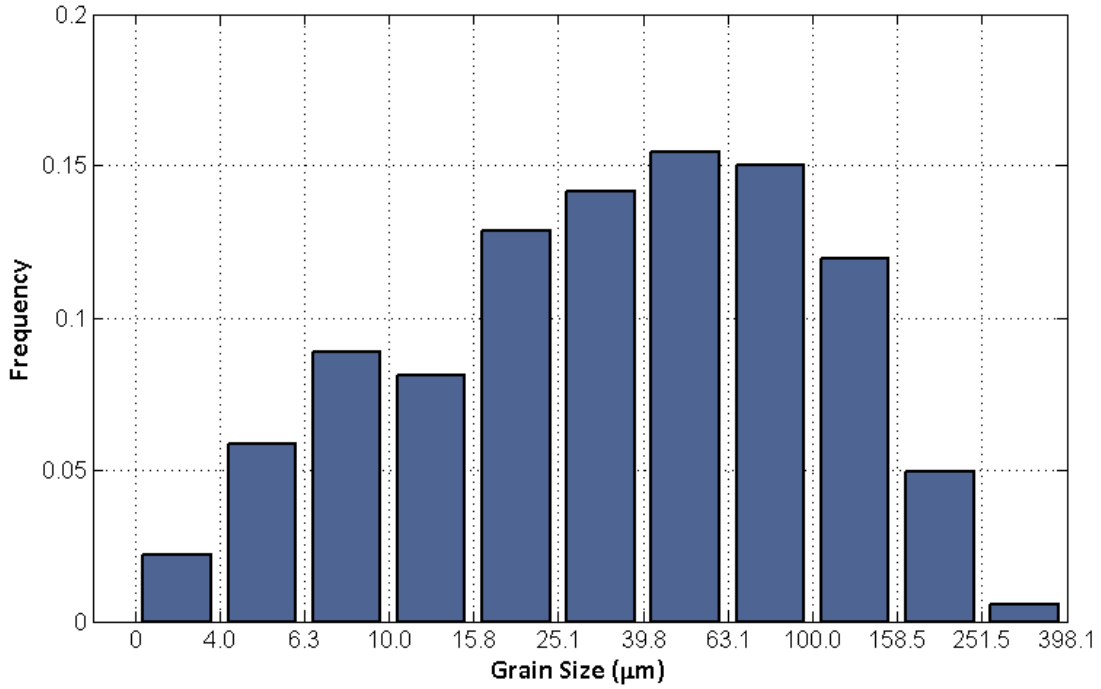


Figure 2.5 Same distribution shown in Figure 2.4 with class intervals defined by log scale.

The lognormal probability density function is represented by Equation 2.13, where the function, f, is log-normally distributed with the mean (M) and standard deviation (D) of the variable's, d, natural logarithm.

$$f(d|D,M) = \frac{1}{dD\sqrt{2\pi}} e^{\frac{-(\ln d - M)^2}{2D^2}}$$

Equation 2.13

M is given by:

$$M = \ln(\overline{d}) - \frac{1}{2}\ln\left(1 + \frac{S^2}{\bar{d}^2}\right)$$

Equation 2.14

where $\bar{d}$ is the average grain size and S is given by:

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(d_i - \bar{d}\right)^2}$$

Equation 2.15

and D is given by:

$$D = \sqrt{\ln\left(1 + \frac{S^2}{\bar{d}^2}\right)}$$

Equation 2.16

Before progressing further it is important to differentiate between the standard deviation, D, as calculated in Equation 2.16, and the standard deviation, S, as given by Equation 2.15. The value D is the standard deviation of the variable's natural logarithm, while the value *S* is the standard deviation of the non-logarithmized values.

Other models have been proposed to describe the distribution of grain sizes. These alternative distributions include the Hillert distribution [41], the Louat distribution [42], and the gamma distribution [43]. Groeber et al [44] compared the effectiveness of the lognormal, Louat, and Hillert distribution models to describe the grain size distribution of IN100. Figure 2.6 shows the equivalent sphere radii (ESR) of 3D reconstructed grains sectioned and imaged using a DB FIB-SEM, and three curves representing the lognormal, Louat, and Hillert distributions. The Hillert distribution shifts the peak to a larger ESR than seen in the experimental data. The Hillert distribution also fails to accurately model the upper tail of the data, while the Louat distribution over-estimates the number of grains at the lower tail. Fátima Vaz and Fortes [43] suggested that because the gamma distribution predicted a larger number of grains at the lower tail compared with lognormal, it would be more appropriate model to describe a grain size distribution. However, it was conceded by Fátima Vaz and Fortes that very little difference existed between the distributions and both were appropriate for describing grain size distribution.

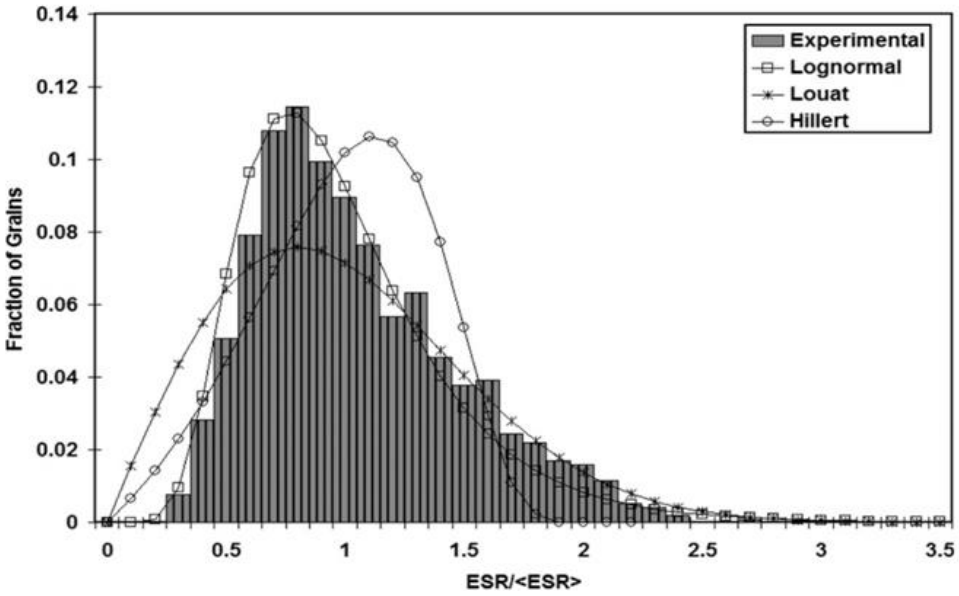


Figure 2.6 Distribution of equivalent sphere radii (ESR), <ESR> = 1.76μm. Curves representing the Hillert, Louat, and lognormal distributions included [44].

2.3 Grain Boundaries

2.3.1 Crystallographic Description of Grain Boundaries

A grain boundary is a surface where two dissimilarly oriented crystals of the same phase meet. The macroscopic grain boundary geometry is described using five degrees of freedom (DOF) to define the misorientation and interface planar normals of adjacent crystals. Grain boundary geometry can be expressed in either the interface-plane scheme or the misorientation scheme. Figure 2.7 is a schematic representation of the interface-plane scheme in which four DOF are required to define the interfacial planar normal, N, for each grain (two DOF per grain) with respect to the local crystal orientation. The final DOF is used to define the twist angle, Φ, between the two lattices.

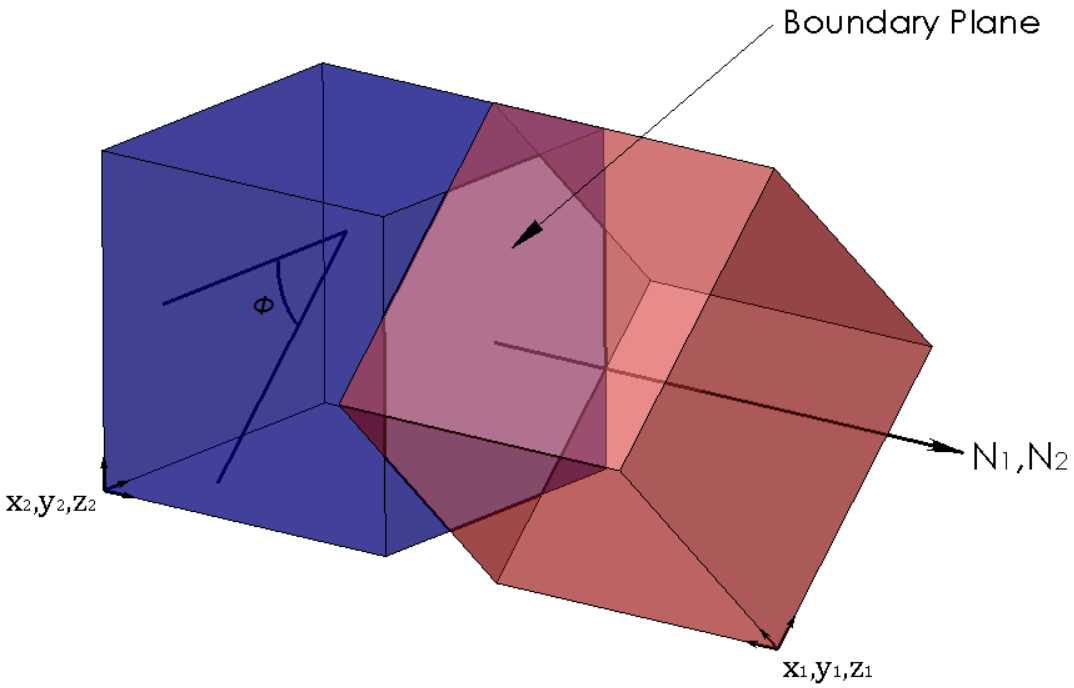


Figure 2.7 Geometry of a grain boundary – interface-plane scheme.For simplicity the boundary normals, N_1 and N_2, are displayed as coincident.

The more commonly used method to define grain boundary geometry is called the misorientation scheme. In this scheme, illustrated in Figure 2.8, the interface between neighbouring grains is described by the relative rotation between orientations of the two neighbouring lattices, leading to an angle/axis pair ($\omega/\hat{n}$). The rotation is accomplished by identifying a direction (axis, $\hat{n}$), such that a rotation (angle, ω) about this axis results in the atoms in each crystal lattice achieving coincidence.

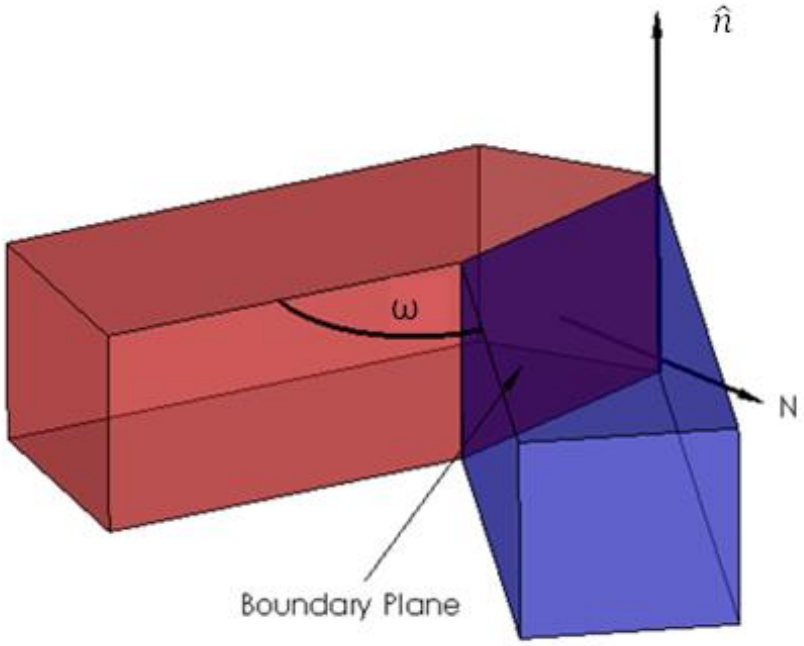


Figure 2.8 Geometry of a grain boundary – misorientation scheme – boundary plane indicated by vector *N* may be inclined at any angle.

The axis of misorientation represents two degrees of freedom and the angle one. Unlike the interface-plane scheme, the misorientation scheme does not fully define the geometry of a grain boundary. The final two degrees of freedom describe the inclination of the grain boundary interface, indicated as N in Figure 2.8.

Grain boundary geometry can be decomposed into two sequential operations – a tilt rotation followed by a twist rotation. While many HGBs tend to be composed of both a tilt and twist operation, there are occasions when boundaries are constructed by simply a twist *or* tilt component. Using the interface-plane scheme, in which a boundary is defined by two planar normals, $h_1k_1l_1$ and $h_2k_2l_2$, and a twist angle, Φ, the following configurations may exist:

1. $h_1k_1l_1 = h_2k_2l_2$ and $\Phi = 0$ – Symmetrical tilt boundary (STB).
2. $h_1k_1l_1 \neq h_2k_2l_2$ and $\Phi = 0$ – Asymmetrical tilt boundary (ATB)
3. $h_1k_1l_1 = h_2k_2l_2$ and $\Phi \neq 0$ – Twist boundary (TWB)

This terminology is particularly relevant to those boundaries that display low interfacial energy and ‘special’ properties.

2.3.2 Representing Crystal Orientation

Euler Angles

Crystal orientation is typically expressed as Euler angles. Euler angles rotate a crystal from a reference coordinate frame by performing three sequential rotations around defined axes. It is important to note that the reference axis rotates after each step. In the Bunge convention, the three rotations are as follows:

1. Around Z axis by angle φ_1.
2. Around X axis by angle Φ.
3. Around Z axis by angle φ_2.

The orientation of the crystal is given as a direction cosine matrix, g:

$$g(\varphi_1, \Phi, \varphi_2) = \begin{bmatrix} \cos\varphi_1 \cos\varphi_2 - \sin\varphi_1 \cos\Phi \sin\varphi_2 & \sin\varphi_1 \cos\varphi_2 + \cos\varphi_1 \cos\Phi \sin\varphi_2 & \sin\Phi \sin\varphi_2 \\ -\cos\varphi_1 \sin\varphi_2 - \sin\varphi_1 \cos\Phi \cos\varphi_2 & -\sin\varphi_1 \sin\varphi_2 + \cos\varphi_1 \cos\Phi \cos\varphi_2 & \sin\Phi \cos\varphi_2 \\ \sin\Phi \sin\varphi_1 & -\sin\Phi \cos\varphi_1 & \cos\Phi \end{bmatrix}$$

Equation 2.17

Quaternions

Quaternions, q, are four-vectors used to describe crystal orientation. Quaternions are comprised of a scalar component, q_0, and an axis component, $\boldsymbol{q}$.

$$q = [q_0, \boldsymbol{q}] = [q_0, q_1 q_2 q_3]$$

Equation 2.18

Euler angles are expressed as a quaternion by Equation 2.19.

$$q = \left[cos\frac{\Phi}{2} cos\frac{\varphi_1 + \varphi_2}{2}, sin\frac{\Phi}{2} cos\frac{\varphi_1 - \varphi_2}{2}, sin\frac{\Phi}{2} sin\frac{\varphi_1 - \varphi_2}{2}, cos\frac{\Phi}{2} sin\frac{\varphi_1 + \varphi_2}{2}\right]$$

Equation 2.19

The advantage quaternions have over direction cosine matrices when performing calculations is immediately recognised. Instead of a matrix of nine elements, the quaternion expresses the same information in four elements, thus improving efficiency when calculating relationships between crystal orientations.

Misorientation between Crystals

The quaternion, q_{mis}, representing the orientation difference (misorientation) between two crystals, q_1 and q_2, is given by Equation 2.20, where ω donates the misorientation angle and $\hat{n}$ the misorientation axis.

$$q_{mis} = q_1^{-1} \cdot q_2 = \left[q_{0,1}q_{0,2} + \boldsymbol{q_1} \cdot \boldsymbol{q_2}, \qquad q_{0,1}\boldsymbol{q}_2 - q_{0,2}\boldsymbol{q}_2 - \boldsymbol{q}_1 \times \boldsymbol{q}_2\right] = \left[\cos\frac{\omega}{2}, \sin\frac{\omega}{2}\hat{n}\right]$$

Equation 2.20

Taking crystal symmetry operators, S_i, into consideration, the *disorientation angle* is calculated. The disorientation angle is defined as the smallest misorientation angle if all 24 symmetry operators for a cubic material are considered. Equation 2.21 gives the quaternion, Δq_{dis},representing the disorientation angle, ω_{dis}, and axis, $\hat{n}$.

$$\Delta q_{dis} = \min_{i=1-24} \Delta q \cdot S_i$$

Equation 2.21

where the 24 symmetry operators, S_i, for the cubic crystal structure are:

$S_1 = (1; 0,0,0)$ $S_2 = (0; 1,0,0)$ $S_3 = (0; 0,1,0)$

$S_4 = (0; 0,0,1)$ $S_5 = (0.5; 0.5,0.5,0.5)$ $S_6 = (0.5; -0.5, -0.5 - ,0.5)$

$S_7 = (0.5; 0.5, -0.5,0.5)$ $S_8 = (0.5; -0.5,0.5, -0.5)$ $S_9 = (0.5; -0.5,0.5,0.5)$

$S_{10} = (0.5; 0.5, -0.5, -0.5)$ $S_{11} = (0.5; -0.5, -0.5,0.5)$ $S_{12} = (0.5; 0.5,0.5, -0.5)$

$S_{13} = \left(\frac{1}{\sqrt{2}}; \frac{1}{\sqrt{2}}, 0,0\right)$ $S_{14} = \left(\frac{1}{\sqrt{2}}; 0, \frac{1}{\sqrt{2}}, 0\right)$ $S_{15} = \left(\frac{1}{\sqrt{2}}; 0,0, \frac{1}{\sqrt{2}}\right)$

$S_{16} = \left(\frac{1}{\sqrt{2}}; -\frac{1}{\sqrt{2}}, 0,0\right)$ $S_{17} = \left(\frac{1}{\sqrt{2}}; 0, -\frac{1}{\sqrt{2}}, 0\right)$ $S_{18} = \left(\frac{1}{\sqrt{2}}; 0,0, -\frac{1}{\sqrt{2}}\right)$

$S_{19} = \left(0; \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$ $S_{20} = \left(0; -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$ $S_{21} = \left(0; 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$

$S_{22} = \left(0; 0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ $S_{23} = \left(0; \frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)$ $S_{24} = \left(0; -\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right)$

From this point on, the term misorientation will be used exclusively, and at all times can be considered the smallest angle necessary to bring two crystals into coincidence by a rotation, ω, around an axis, $\hat{n}$.

2.3.3 Measuring Grain Boundary Geometry

The geometry of a grain boundary can be measured using a variety of experimental methods:

1. X-ray diffraction
2. Transmission electron microscope (TEM)
    i. Selected area diffraction (SAD)
    ii. Convergent beam electron diffraction (CBED)
3. Scanning electron microscope (SEM)
    i. Electron Backscatter Diffraction (EBSD)

By far the most commonly employed tool for determining grain boundary geometry is EBSD. This is predominantly due to the relatively quick material preparation procedures and the ability to collect large amounts of data automatically.

Without the assistance of EBSD to provide a crystallographic description of boundaries, typically morphological indicators are used to identify boundary types. The identification of twins is often performed by observing boundary morphology from the optical micrograph or SEM image, and as a consequence the method is susceptible to operator bias. Observations from EBSD mapping and 3D studies presented in this thesis indicate that 2D morphology observed on optical micrographs suggesting a twin boundary, may be the result from sectioning other boundary types.

Electron Backscatter Diffraction (EBSD)

EBSD operates by positioning a flat, polished sample at an angle (70°) to the electron beam in a scanning electron microscope. When the beam interacts with the crystal lattice, low energy backscattered electrons are projected at angles determined by the crystal lattice parameters. These interactions lead to constructive and destructive interference. When a phosphor screen is placed a short distance from the tilted sample a diffraction pattern is observed. The diffraction pattern, often referred to as a Kikuchi pattern, is imaged from the phosphor screen using a low light camera. A schematic of the EBSD setup is shown in Figure 2.9.

The Kikuchi pattern obtained is a function of the crystal orientation and lattice parameters of the material being analysed. After first identifying the phase, EBSD software then indexes the pattern. Indexing involves the software automatically locating the positions of individual Kikuchi bands, comparing these to theoretical data about the relevant phase, and rapidly calculating the

crystallographic orientation. An example of an indexed pattern is shown in Figure 2.10 for the austenite phase in Alloy 800H.

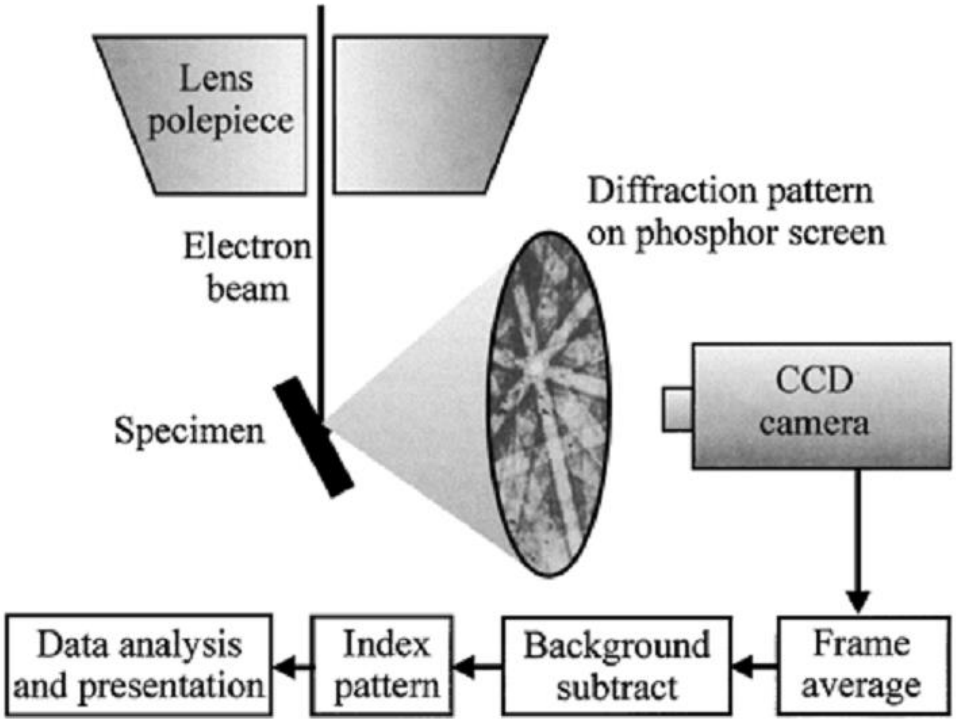


Figure 2.9 Schematic diagram of typical EBSD setup in the SEM illustrating a diffraction pattern produced from a tilted specimen and captured by a CCD camera [45].

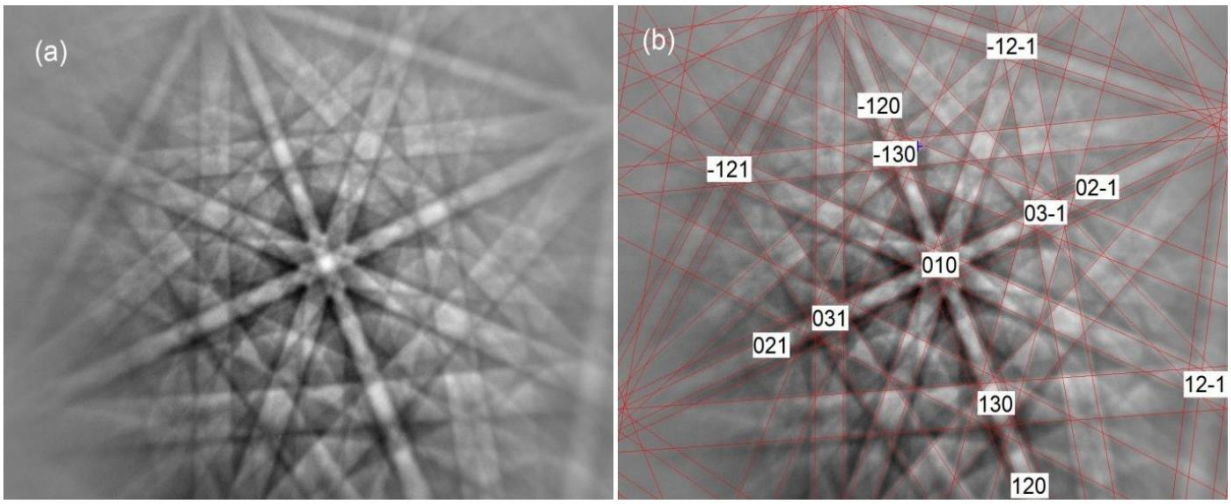


Figure 2.10 EBSD pattern indexing for the austenite phase. (a) Unindexed Kikuchi diffraction pattern. (b) Pattern indexed as austenite with zones labelled.

To create a grain boundary map, the beam is “stepped” across the sample surface in a user-defined grid. The orientation is calculated at each grid point to a precision within 0.5-1.0° [45]. The misorientation between any two grid points is calculated, and if greater than 15° a HGB is said to exist between the adjacent pixels. By repeating this process for all positions on the grid, a grain boundary map is produced, an example of which is displayed in Figure 2.11**.** EBSD mapping is used to calculate data such as grain size statistics. Furthermore, because the misorientation across each grain boundary segment is available from EBSD data, three of the five degrees of freedom necessary to fully define grain boundary geometry are available.

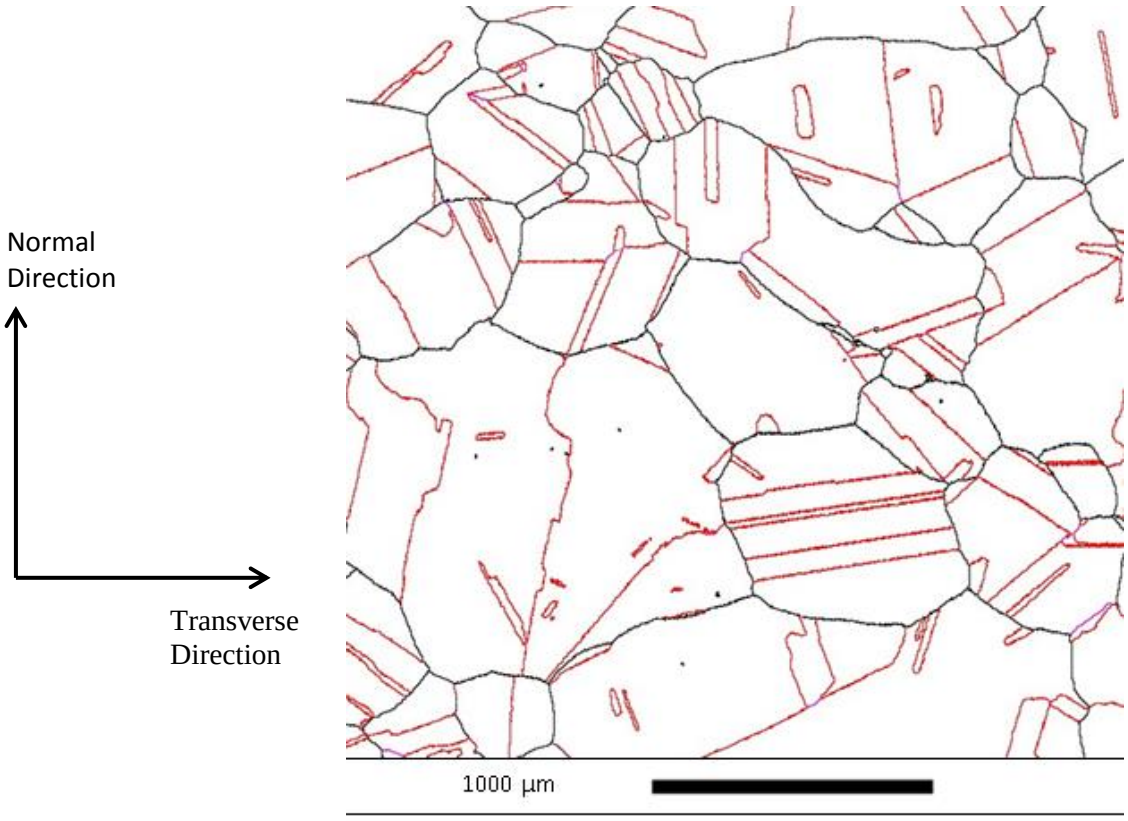


Figure 2.11 Grain boundary map of Alloy 800H produced using EBSD. Red boundaries were identified as having 60°/111 misorientation and black boundaries represent all other high angle grain boundaries.

2.3.4 Grain Boundary Types

Low-Angle Grain Boundaries

If the misorientation between adjoining grains is small, typically less than 10 - 15° [29], a low-angle grain boundary (LGB) exists. The structure of a LGB can be described as an array of lattice dislocations [46], illustrated in Figure 2.12 for a symmetric tilt boundary composed of parallel edge dislocations.

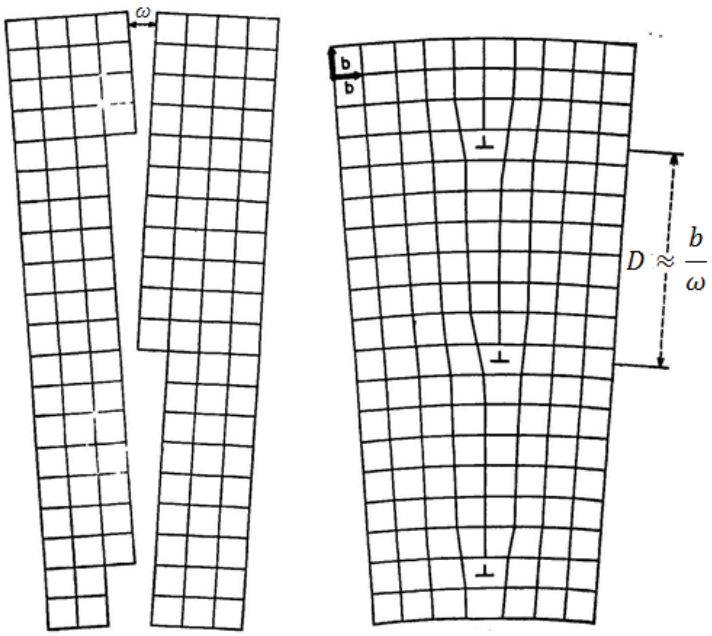


Figure 2.12 A low-angle tilt boundary composed of edge dislocations with ω misorientation [47].

For a LGB, the misorienation angle, ω, is related to the size of the Burgers vector, b, and the dislocation spacing, D, by the expression:

$$\omega = \frac{b}{D}$$

Equation 2.22

The energy, γ, of a low-angle grain boundary can be calculated by the Read-Shockley dislocation model [48]:

$$\gamma = \gamma_0 \omega (A - \ln \omega)$$

Equation 2.23

where γ_0 and A are constants, and ω is the misorientation angle between two grains.

Figure 2.13 is the schematic curve representing the energy of a low-angle grain boundary as a function of misorientation. As the spacing between dislocations decreases with increased boundary misorientation, the strain fields of the dislocations cancel out so that γ increases at a decreasing rate. When ω exceeds 10-15°, the dislocation spacing is close enough for dislocation cores to overlap making it difficult to identify the individual dislocations.

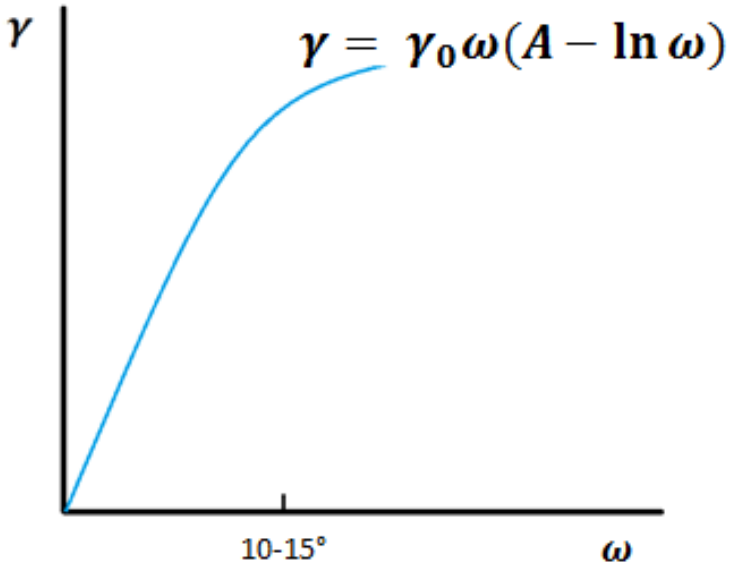


Figure 2.13 Schematic of the energy, γ, of a low-angle grain boundary as a function of misorientation, ω, according to Equation 2.23.

High-Angle Grain Boundaries

The transition angle from low-angle to high-angle grain boundary (HGB) depends on the material, the crystal structure, and the boundary structure. HGBs may be distinguished from LGBs based on atomic configurations in the vicinity of the grain boundary [49], grain boundary free energy [50], or the activation enthalpy for boundary migration [51]. The atomic arrangements within a HGB may be described using concepts such as the coincident site lattice (CSL) model [52].

The coincident site lattice (CSL) was first described in 1949 [52], and represents a method of classifying certain types of grain boundaries which have the potential to exhibit low energy. Certain rotations, ω, around certain axes, $\hat{n}$, result in the superposition of a proportion of lattice points from each lattice – a coincident site lattice.

To understand the CSL notation, consider two interpenetrating lattices which make a theoretical 'superlattice', where the atomic positions of one lattice are superimposed onto the atomic positions of the other lattice. This is shown in Figure 2.14 in two dimensions.

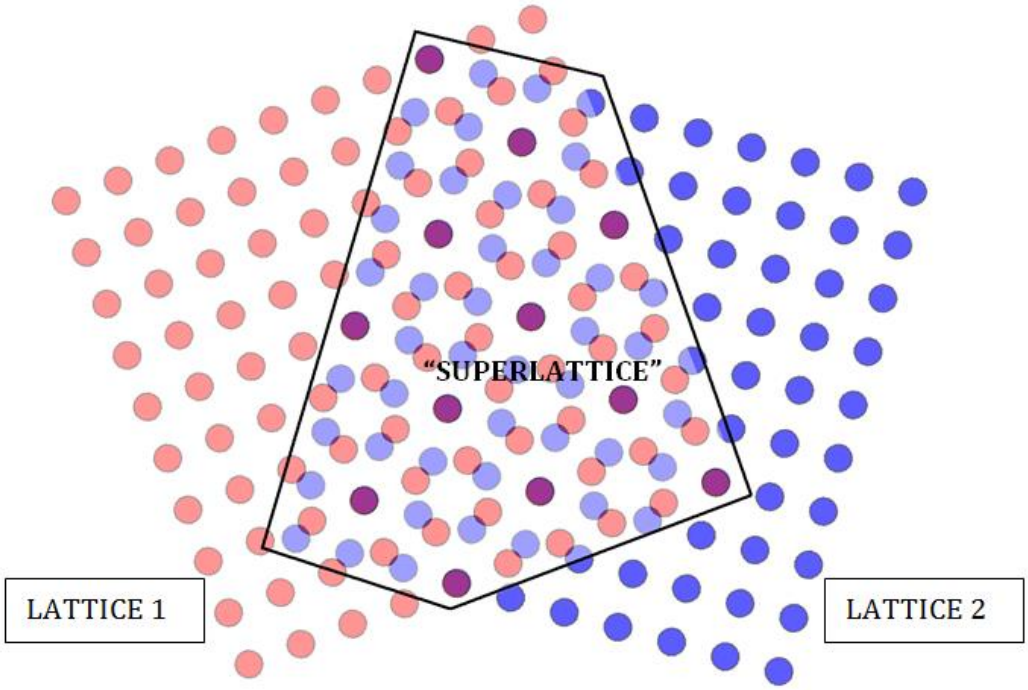


Figure 2.14 Theoretical 'superlattice' illustrating the CSL model. Purple sites indicate where atoms are coincident. In this case one in five sites is coincident, indicating a Σ5 boundary.

Low Σ values and their misorientations for cubic materials are listed Table 2.1. Also listed are the maximum deviations, v_m, from the ideal misorientation. This is based on the Brandon criterion [49], given by Equation 2.24 where Σ is the is the inverse of coincident sites:

$$v_m = 15\,\Sigma^{-\frac{1}{2}}$$

Equation 2.24

The Brandon criterion is based on the idea that certain deviations from the exact CSL misorientation can be accommodated by arrays of dislocations. Other, generally more restrictive criteria such as the Palumbo and Aust criterion [53] are sometimes used. For comparison, for the Σ3 boundary, the Palumbo and Aust criterion allows a maximum deviation of 6.0° compared to the 8.67° allowed by the Brandon criterion. In the current study, it was observed that the Palumbo and Aust criterion detected over 95% of the Σ3 boundaries identified by the Brandon criterion. The Brandon criterion is by far the most commonly reported in the literature, and, due to both criterions identifying approximately the same amount of Σ3 boundaries, the Brandon criterion will be used in the current study to maintain continuity.

Table 2.1 Specific lattice misorientations for Σ3 to Σ27 in cubic crystals.

Σ Value	Misorientation Angle, ω	Misorientation Axis, $\hat{n}$	Deviation, v_m
3	60°	<111>	8.67°
5	36.87°	<100>	6.71°
7	38.21°	<111>	5.67°
9	38.94°	<110>	5.00°
11	50.48°	<110>	4.52°
13a	22.62°	<100>	4.16°
13b	27.80°	<111>	4.16°
15	48.19°	<210>	3.87°
17a	28.07°	<100>	3.64°
17b	61.93°	<221>	3.64°
19a	26.53°	<110>	3.44°
19b	46.83°	<111>	3.44°
21a	21.79°	<211>	3.27°
21b	44.40°	<311>	3.27°
23	40.45°	<100>	3.13°
25a	16.25°	<331>	3.00°
25b	51.68°	<110>	3.00°
27a	31.58°	<100>	2.89°
27b	35.42°	<221>	2.89°

Other models used to describe the structure of high-angle grain boundaries include the structural unit model [54]. According to the structural unit model, a high-angle grain boundary can be geometrically described by repeated units representing the arrangement of atomic positions at the grain boundary. Ashby [54] describes eight different structural units, and suggests that boundaries with a high number of coincident sites can be described by repeating a single unit. While the structural unit model may also be used to identify potentially low energy grain boundaries, the CSL model has the advantage of being easily applied to EBSD data because boundary type is defined by the measured misorientation.

Because the CSL model and Brandon criterion will be used to define grain boundary types, for consistency it will also be used to define the misorientation at which a LGB transitions to a HGB. A Σ1 describes a boundary with atomic positions of adjacent crystal structures with coincident in a 1:1 ratio.

Using the Brandon criterion (Equation 2.24), a Σ1 boundary may describe all boundaries with misorientations 0° - 15°. Therefore, in the current study HGBs will include all misorientations greater than 15°.

The interfacial energy of HGBs can be three to six times higher than low-angle grain boundary energy [55]. No simple relationship exists between the energy of a boundary and its overall geometry as defined by its degrees of freedom [56]. While the CSL model suggests that boundaries with a high number of coincident sites would have lower energies, simulations using the embedded atom method (EAM) performed by Rittner and Seidman [57] indicate that this is not always the case. The study simulated 21 <110> symmetric tilt boundaries (Table 2.2), and results (Figure 2.15) showed no correlation between coincident site density and energy. For example, three of the boundaries studied were Σ3, but only the boundary with interface plane (1 1 1) showed an energy cusp. Two of the Σ3 boundaries had energies similar to the other boundaries studied and the only other energy cusp was produced by a Σ11 boundary.

Table 2.2 Σ value, tilt angle, and boundary plane for the 21 grain boundaries.

Σ Value	<110> Tilt Angle	Boundary Plane	Σ Value	<110> Tilt Angle	Boundary Plane	Σ Value	<110> Tilt Angle	Boundary Plane
73	13.44°	(1 1 12)	33	58.99°	(2 2 5)	33	121.01°	(5 5 4)
33	20.05°	(1 1 8)	3	70.53°	(1 1 2)	11	129.52°	(3 3 2)
19	26.53°	(1 1 6)	43	80.63°	(3 3 5)	9	141.06°	(2 2 1)
27	31.59°	(1 1 5)	17	86.63°	(2 2 3)	27	148.41°	(5 5 2)
9	38.94°	(1 1 4)	17	93.37°	(3 3 4)	19	153.47°	(3 3 1)
57	44.00°	(2 2 7)	43	99.37°	(5 5 6)	3	159.95°	(4 4 1)
11	50.48°	(1 1 3)	3	109.47°	(1 1 1)	73	166.56°	(6 6 1)

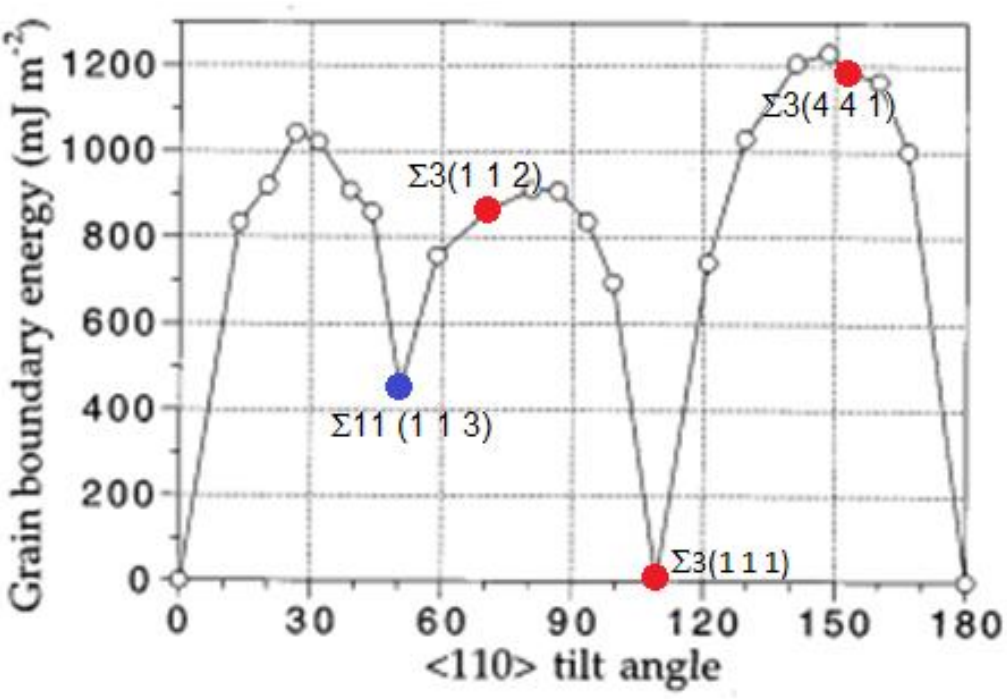


Figure 2.15 Computed grain boundary energies for 21 <110> symmetric tilt boundaries as a function of tilt angle.

The result from the study highlights a major limitation of the CSL model. While a boundary requires five DOF in order to be fully defined, the CSL model is based on misorientation alone, which only provides three DOF. The use of the CSL model to correlate boundary structure to properties is lacking, due to its inability to fully describe the boundary. In order to improve structure-property correlation, one must also consider the interface plane.

2.4 Twin Boundaries

When plastically deformed, with or without subsequent annealing, many metals produce twin boundaries. Twins formed during straining are called mechanical twins, and those formed during annealing (recrystallization), annealing twins. Because this body of work deals exclusively with annealing twins, from this point onwards no distinction shall be made between the two. Often the term 'twin boundary' is incorrectly used to describe any boundary with Σ3 misorientation (e.g. [58, 59]), and occasionally boundaries with any $\Sigma 3^n$ (where $n \leq 3$) misorientation will be described as twin variants [60], higher order twins [61], or twin related rotations [62].

There are several configurations in which the Σ3 boundary category may be subdivided. To avoid confusion, in the current study only the symmetrical tilt boundaries will be referred to as twins:

1. The {111/111} symmetrical tilt boundary – coherent twin
2. The {211/211} symmetrical tilt boundary – incoherent twin
3. Asymmetrical tilt boundaries (ATB) on the <110> zone – non-coherent Σ3
4. Other asymmetrical tilt boundaries and twist boundaries – non-coherent Σ3

The boundary energies of the coherent twin and a random HGB in 304 stainless steel were determined as 19mJ/m^2 and 835mJ/m^2 respectively [40]. The asymmetrical tilt boundaries on the <110> zone are significant because they display lower energies (10mJ/m^2-610mJ/m^2 for copper [63]) than other boundaries in the Σ3 classification.

2.4.1 Twin Boundary Morphology

Figure 2.16 illustrates the commonly observed 2D twin/Σ3 morphologies: one sided twins, complete parallel-sided twins, incomplete parallel-sided twins, and island twins [64]. In many cases, twin boundaries have some degree of faceting observable with optical microscopy [65]. Examples of twin boundaries with faceting are shown in Figure 2.17. Researchers [66-71] identified that the facets along the length of twins, Figure 2.17(a), and at the end of incomplete parallel-sided twins, Figure 2.17(b), typically have boundary plane indices categorising them as ATBs on the <110> zone.

Norbygaard [72] identified that faceting was prevalent on Σ3 boundaries close to the ideal 60°/<111> misorientation, i.e. on Σ3 boundaries considered more likely to be coherent twins [59]. Therefore, the dissociation of a Σ3 into boundary with a broad {111} (coherent twin) interface with low energy ATB facets is favourable, assuming the increase in the total boundary area is overcome by the decrease in

the interfacial energy [50]. Because the coherent twin has energy on the order of 30 times less than other Σ3 boundaries, the dissociation of Σ3 boundaries should be favourable. In the current study, faceting was observed on the majority of twin boundaries.

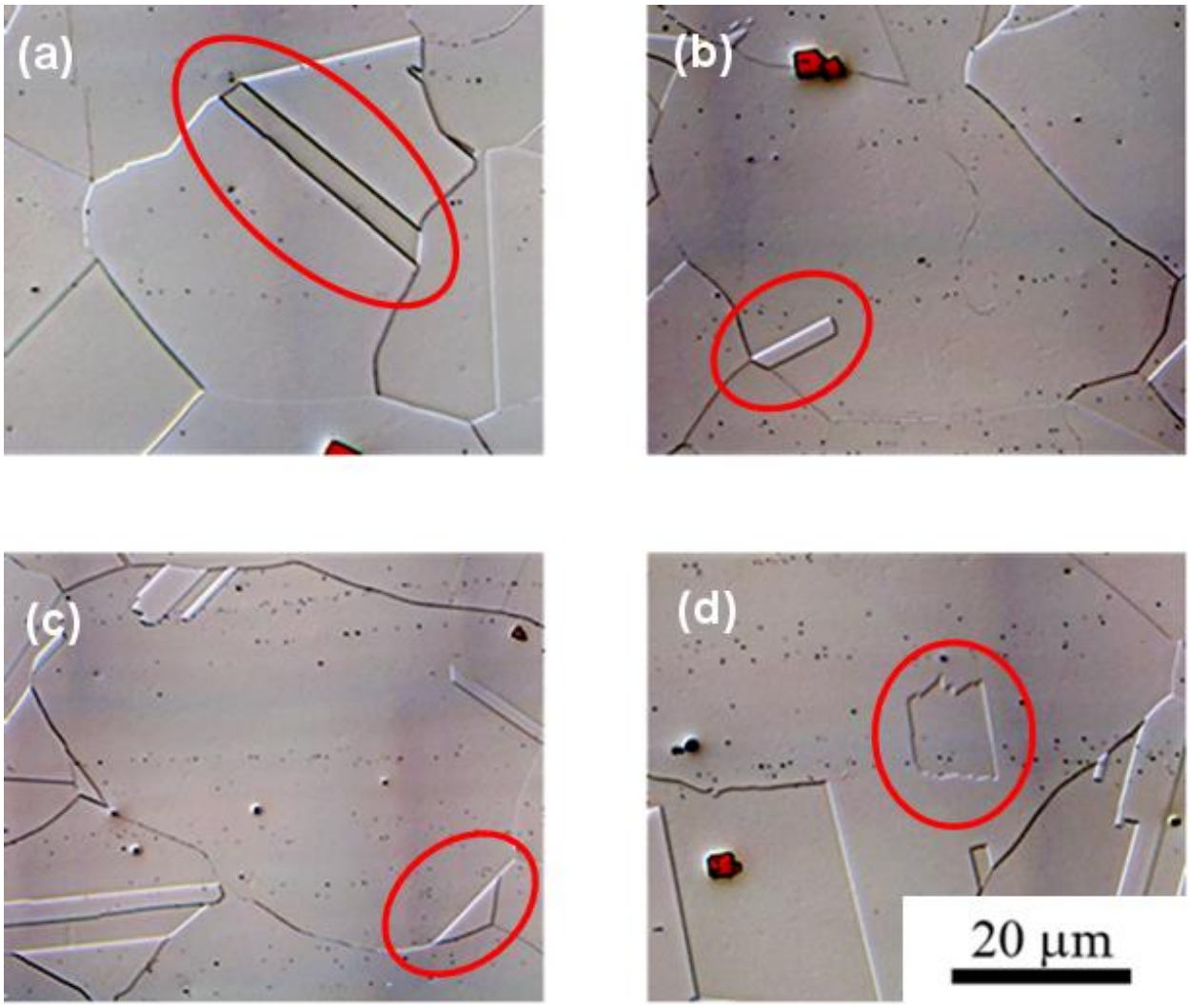


Figure 2.16 (a) Complete parallel sided twin (b) incomplete parallel sided twin (c) one sided twin (d) island twin.

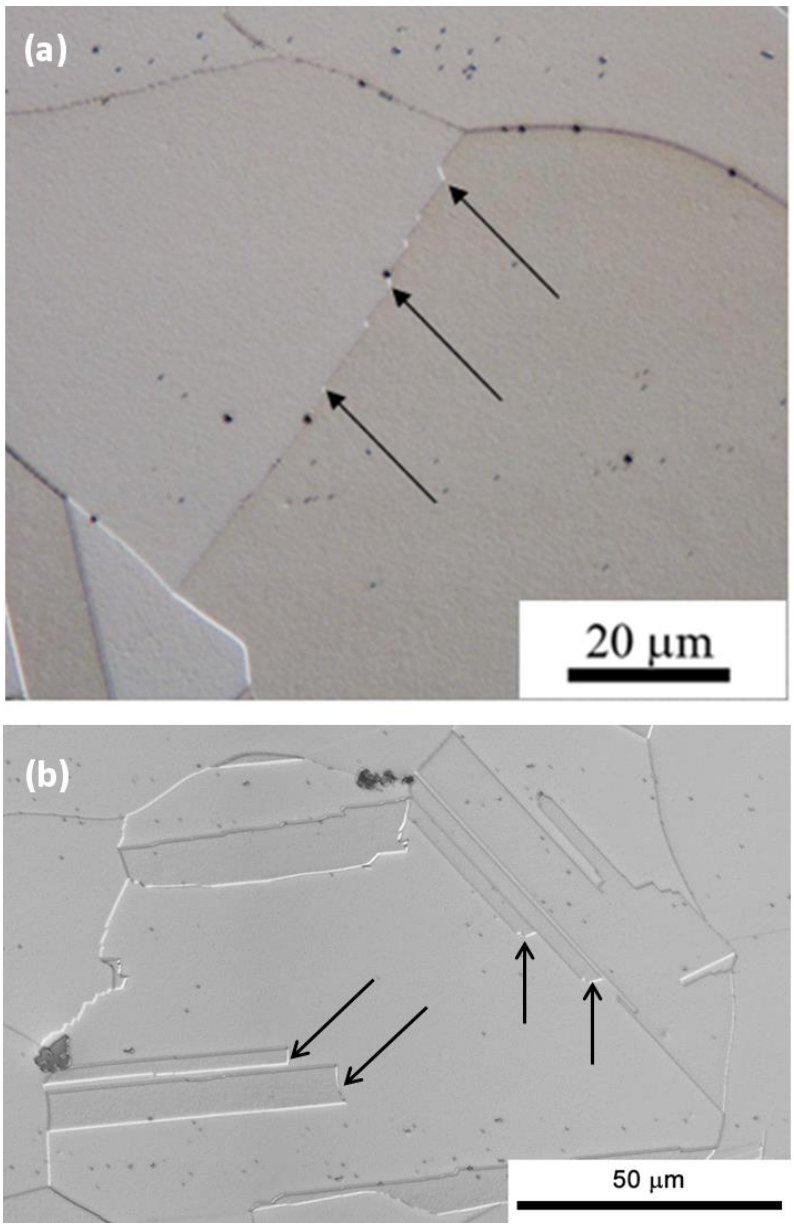


Figure 2.17 Faceting on boundaries indicated by arrows. (a) Faceting along a boundary (a) faceting at the end of incomplete parallel twins.

Research [64, 73] suggests that morphologies observed on 2D surfaces are the result of the 3D morphology and the orientation of the sectioning plane. The reconstruction of serial sectioning data allows for the visualization of the 3D structure of opaque materials. Serial sectioning is comprised of three main tasks: sectioning, data collection, and feature segmentation.

Sectioning involves the removal of a known thickness of material. Focused ion beam (FIB) milling [74], mechanical polishing [75], and micromilling [76] are examples of sectioning techniques. The sectioning thickness is usually determined by the size of the microstructural feature to be examined. A general rule is to acquire a minimum of ten sections per feature [77].

After sectioning, the surface of interest is characterised through any number of techniques. Optical microscopy offers the ability to collect data easily, and if equipped with a motorized stage and autofocus capabilities (e.g. Robo-Met.3D [78]) image montages can be created with limited operator input. A limitation of optical imagery is the segmentation of the features of interest. Automated segmentation relies on sufficient contrast for computer algorithms to distinguish between features. For example, Kral and Spanos [79] were able to successfully segment proeutectoid cementite from an austenite matrix, while automated segmentation performed by Lewis et al [75] on a stainless steel (AL-6XN) required manual intervention to identify some of the grain boundaries.

Feature segmentation is aided by the use of EBSD and EDS mapping [80]. Phase and crystal orientation information is used to group pixels and identify boundaries. Reed [1] produced 42 serial sectioned EBSD maps from 304 stainless steel at a step size (mapping resolution) of 5µm. The final dimensions of the sectioned volume were 2300x2300x267µm^3. The purpose of the study was to compare the 3D and 2D number fractions of different CSL boundaries. EBSD mapping was ideal for this investigation because there was no requirement to produce images with resolution comparable to an optical microscope, i.e., fine boundary detail was not important. The obvious limitation of EBSD is time. It takes approximately 80 minutes (~1000 points/second [81]) to produce an orientation map with the same resolution as an optical micrograph (~5Mpx).

A dual beam FIB-SEM can perform the sectioning and EBSD mapping steps automatically within an SEM chamber. Uchic et al [74] performed serial sectioning on a nickel based alloy (IN100) using a FIB-SEM producing a 50x50x50µm^3 volume in four days. The EBSD maps were produced with a step size of 0.25µm, and FIB milling removed material at a depth of 0.25µm. While automation is a benefit of the FIB-SEM technique, the limitation again is the EBSD mapping time required to produce large volumes of sectioned data.

Lewis et al [75] employed both optical microscopy and EBSD to collect microstructural data, producing a serial section volume of approximately 250x250x160µm^3 (48 sections). While all sections were imaged optically, only every tenth section was mapped using EBSD. Optical micrographs were used to identify the location of the grain boundaries and EBSD was used to measure the orientations of the grains. Therefore, for each sectioned grain, a complete 3D morphological and crystallographic characterisation could be made. The resulting 3D reconstructions allowed for the determination of the grain boundary plane inclination, and, along with the orientation data, all five DOF which define a grain boundary were determined.

Bystrzycki et al [73] studied the 3D morphology of twin boundaries in a $NiMn_2$ sample with an average grain size of 200µm. 100 serial sections (8µm depth per section) were imaged optically at 100x magnification and 3D models of the grains reconstructed in Auto CAD. In the study, a grain was defined by a volume bounded by non-twin boundaries and volumes bounded by at least one twin boundary were called twins.

Measurements of the twin boundary surface area, S_{TB}, number of twin boundaries per grain, N_{TB}, and the volume of a grain, V_G, were performed for 30 grains containing twins. The results showed that while the boundary surface area scaled linearly with grain volume, the number of twins per grain did not. This result suggests that twin boundaries form during the initial stages of recrystallization and grow as the grain coarsens, but grain coarsening does not promote further twin formation.

The 3D morphology of twins was also analysed, concluding that twins can be categorised as either lamella twins or edge twins. Lamella twins are characterised by the existence of two parallel twinning planes. A section taken through a lamella twin volume can produce any of the four 2D morphologies shown in Figure 2.16. An edge twin only had one twin interface so that sectioning would always produce the 2D morphology described as a one-sided twin. Although sections would produce 2D morphologies described as an island twin, at all times it was shown to be part of a volume connected to the boundary network.

Randle et al published papers measuring the grain boundary plane indices of Σ3 boundaries for Ni [82] and Cu [83, 84]. The procedure [85] used to measure boundary plane indices was the same for all four studies. A polished and etched section was examined in an SEM to identify grain boundaries. EBSD was used to measure the misorientation across selected grain boundaries. Only grain boundaries that produced straight traces on the section were selected for analysis because they were considered more likely to have a planar interface. Approximately 20µm of material was removed by grinding and polishing and the SEM used again to examine the new location of the selected boundaries. Hardness indents were used to calculate section depth and align the parallel sections. Using the boundary positions on the two sections, the inclinations of the interface planes for the selected boundaries were calculated to within ±5°. The interface inclinations combined with the orientation data allowed the planar indices to be calculated.

Table 2.3 summarizes the number of Σ3 boundaries analysed for each sample and its processing history. For each sample, the proportion of the Σ3 population with planar indices indicating tilt boundaries (TB) on the <110> zone was approximately 90%. Σ3 TBs on the <110> zone typically display lower energies than other Σ3 boundaries. The grain boundary energies for copper [63] are shown in Figure 2.18 along

with the interface plane distribution for the Σ3s identified as TBs on the <110> zone. The energy is lowest (0.01J/m^2) for the coherent twin, identified as having {111} interfaces for both grains. As the boundary plane rotates away from the {111} plane around the 110 axis, the energy increases to 0.61J/m^2. This maximum occurs for ATBs with planes 111/511, or equivalently a 71° tilt away from the coherent twin. After a dip, the energy continues to increase up to 0.54J/m^2 for the symmetric incoherent twin, identified as having {211} interfaces for both grains.

Table 2.3 Number of boundaries analysed for nickel and copper for the three anneals. Adapted from [83, 84, 86].

Material	Processing	Number of Σ3 Analysed	Percentage of Σ3s identified as TB on the <110> Zone
Ni	2 hours @ 1000°C + 67.5 hours @ 850°C	92	87%
Cu	1 hour @ 900°C	207	89%
	1 hour @ 900°C + 97 hours @ 540°C	300	90%

Figure 2.18 shows that for all three samples the majority of the Σ3 boundaries studied were identified as either coherent twins or had planes slightly deviated away from being coherent. For example, a 15° deviation away from the coherent twin produces boundary plane combination 221/744. For nickel and copper (1 hour @ 900°C) respectively, 66% and 61% of Σ3 boundaries were identified as have boundary planes representing TB with deviations of 0°-15° from the coherent twin. For copper, this value increased to 87%, including an approximate five times increase in coherent twins, after a further anneal at 540°C for 97 hours. Randle concluded that at increased temperature and given enough time, Σ3 boundaries dissociate achieving a configuration with decreased interfacial energy.

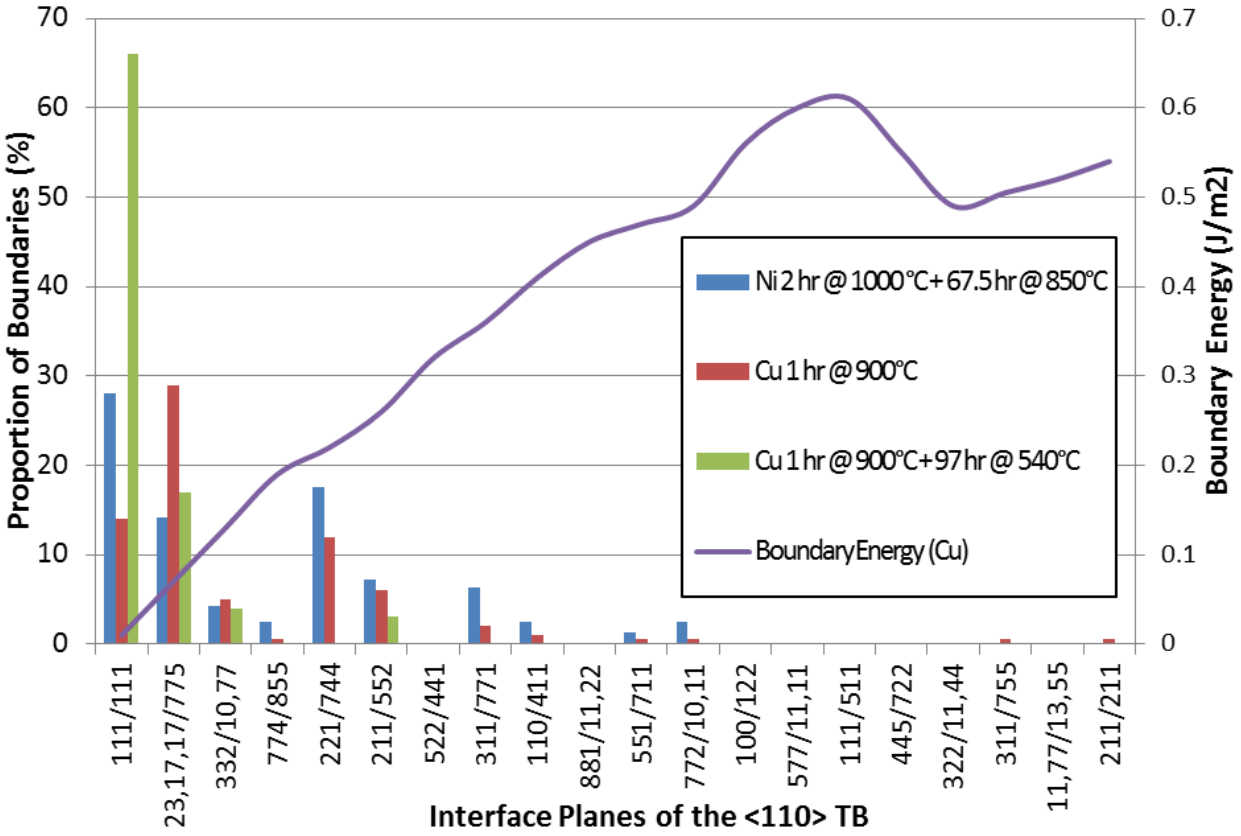


Figure 2.18 Σ3 boundary plane distribution for the boundaries identified as TBs on the <110> zone (adapted from [82-84]). Boundary energy for copper adapted from [63].

A major implication of the study suggests that it cannot be assumed all straight line boundary traces on 2D sections with a Σ3 misorientation are coherent twins. While the description will certainly identify Σ3 boundary with energies significantly less than random HGBs (1J/m^2 for copper), information about the interface plane is required to correctly identify a coherent twin.

While the 3D reconstruction of serial sections provides one way of determining the boundary plane indices, single-surface trace analysis provides a method of identifying possible coherent twins in a less laborious manner. The boundary trace direction plus the misorientation between adjacent grains provides four out of the five degrees of freedom describing a boundary. The boundary trace direction vector, $\hat{l}$, can be measured from the surface of the planar section. Because $\hat{l}$ lies in the grain boundary plane it is orthogonal to the boundary plane normal vector, $\hat{t}$, i.e., $\hat{l} \cdot \hat{t} = 0$. This condition can be used to check if the boundary plane normal could be <111> in both adjacent grains. If $\hat{l} \cdot \langle 111 \rangle \neq 0$ then the boundary plane cannot be {111}, but if $\hat{l} \cdot \langle 111 \rangle = 0$, then the boundary plane might be {111} and therefore may be a coherent twin. Randle [87] validated the trace analysis process showing that only 10% of the cases were ambiguous in terms of recognising a coherent Σ3.

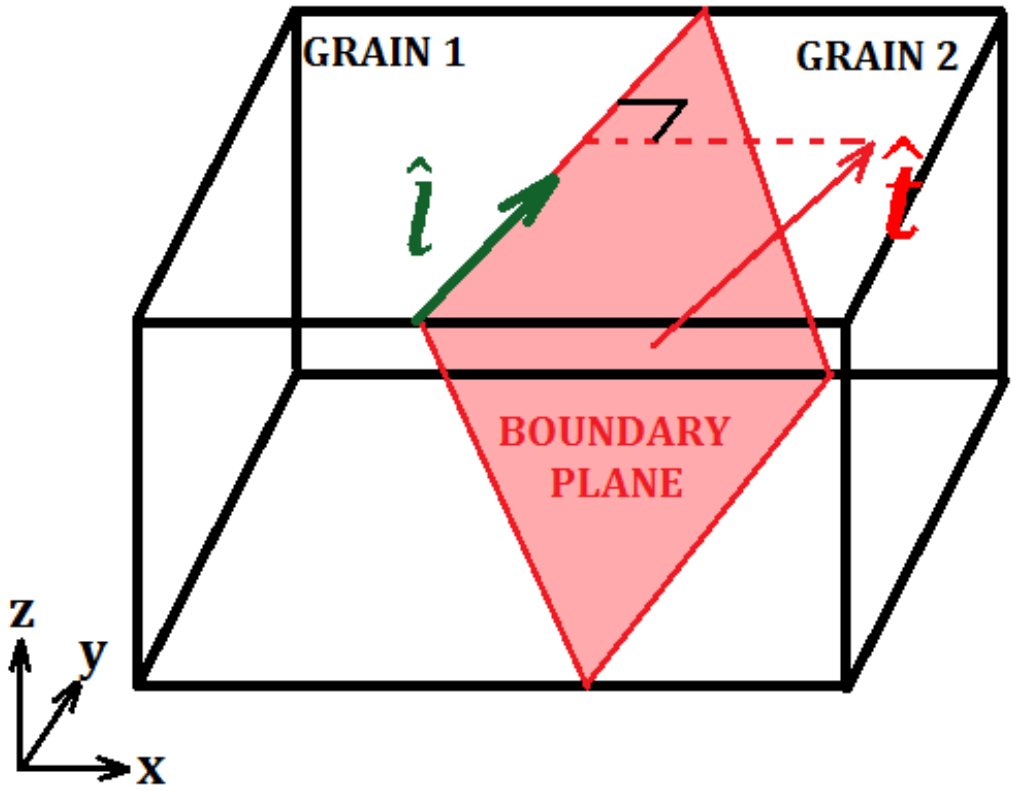


Figure 2.19 Boundary plane with inclination described by the normal vector, $\hat{t}$. The boundary trace produced by the boundary intersecting the sectioning plane is given as $\hat{l}$.

2.4.2 Twin Formation Theories

Prior to the 1950's very little was published in relation to the formation of twins, although several researchers had remarked on the conditions necessary for them to form. In 1926, Carpenter and Tamura [88] reviewed many of the initial studies regarding the formation of twins and made the following observations:

1. Materials that are most likely to form twins are those with face-centred cubic, tetrahedral cubic (diamond cubic), and face-centred tetragonal crystal lattice arrangements.
2. Twins develop by boundary migration.
3. Strain prior to annealing is required to form twins.

Carpenter and Tamura conducted simple experiments on cast copper and cast silver. The copper was cooled slowly after melting to avoid strains, then annealed at 1000°C for three hours but produced no

twins. This same result was obtained with cast silver, which had been shown to produce twins more readily than copper. The silver was dropped on a wooden floor and annealed again with results showing twins formed in the region where the sample impacted the floor.

In 1950, Burke [89] proposed the "growth accident model" for the formation of twins and drew many of the same conclusions summarised by Carpenter and Tamura [88]. One important difference Burke suggested was that deformation was not necessary to produce twins. The observation that a twin could form without the assistance of deformation, that is, a twin is formed simply by grain coarsening, was later dismissed by Neilson [90]. Neilson suggested that the twin observed 'forming' at the corner of a growing grain could in fact have already existed within the grain volume below the sectioning plane. A further point made by Burke, that was supported in subsequent investigations by Fullman and Fisher [91] and Viswanathan [92], is that new grain orientations must be energetically favourable, i.e., the formation of a twin must produce lower total interfacial energy than the previous configuration.

The formation of a twin by the growth accident model is shown in Figure 2.20. Figure 2.20(a) shows the migration of two grain boundaries to the right. If a growth accident occurs, a twin is formed having energy of γ_{tb}, Figure 2.20(b). If the energy balance shown in Equation 2.25 holds true then the twin is stable.

$$\gamma_{tb}A_{12} + \gamma'_{13}A_{13} + \gamma'_{23}A_{23} < \gamma_{13}A_{13} + \gamma_{23}A_{23}$$

Equation 2.25

where A is the grain boundary area, γ is the grain boundary energy, and γ' is the grain boundary energy after the formation of the twins. Figure 2.20(c) and Figure 2.20(d) illustrate that with further boundary migration another twin may form parallel to the first.

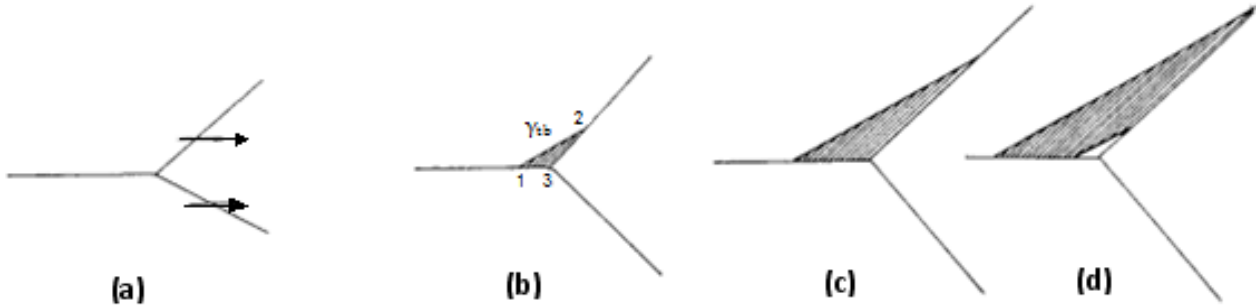


Figure 2.20 The formation of twin formation in the growth accident model [64].

Gleiter [93] proposed an atomistic model based on the idea that annealing twins form due to growth accidents via stacking faults on the {111} planes of coarsening grains. During grain growth, atoms deposit at the steps of the growing grain, and if an incorrect stacking sequence occurs at the growing interface a twin will form. Using the proposed atomistic model, Gleiter derived Equation 2.26 for the probability, p, of a stacking fault occurring on the growing interface (from Equation 8 in [93]).

$$p = exp\left\{\frac{\sigma_Z\left(Q - kT\ln\frac{\Delta G^0}{kT}\right)}{\left[kT\sigma_Z - \frac{\pi kT\varepsilon^2 h^2}{\left(Q - kT\ln\frac{\Delta G^0}{kT}\right)}\right]}\right\}$$

Equation 2.26

$$\delta = 30\gamma$$

Equation 2.27

$$\varepsilon = \frac{1}{2}\delta$$

Equation 2.28

$$\sigma_Z = \frac{1}{2}\gamma$$

Equation 2.29

$$\Delta G^0 = \frac{4\delta}{d}$$

Equation 2.30

where Q is the activation energy for grain boundary migration, ΔG^0 is the difference in the Gibbs' free energy between the growing and the shrinking grains, σ_0 is the surface energy of a coherent twin boundary, h is the height of the step formed by the twin nucleus (taken as the distance between {111} planes), ε is energy of the step, k is the Boltzmann constant, T is absolute temperature, d is the grain diameter, δ is the grain boundary energy, and γ is the stacking fault energy.

Bäro and Gleiter [94] measured the twin density (twins intersecting a straight line on the surface) for Cu-3wt.%Al samples annealed at various temperatures all having an average grain size of 300µm. The experimental data is shown in Figure 2.21 with the twinning probability (proportional to twin density), p, plotted against annealing temperature for seven samples. The black line represents the fit using data provided by Bäro and Gleiter [94] (Table 2.4, and Equation 2.26). The black line is able to predict twinning probability for annealing temperatures below 600°C, but it is not accurate for higher temperatures.

Li et al [95] noted that the stacking fault energy, γ, for Cu-3wt.%Al is approximately 0.020J/m^2, therefore the empirical relationships given by Equations 2.27, 2.28, and 2.30 provided by Gleiter [93] do not hold for δ, ε, and ΔG^0, respectively. Li et al [95] formulated new values for δ, ε, and ΔG^0using γ=0.020J/m^2 and Equations 2.27, 2.28, and 2.30. The twinning probability is shown as a red curve in Figure 2.21. The

result provided by Li et al [95] shows that calculated values for twin density are essentially independent of temperature, an observation also made by Pande [96].

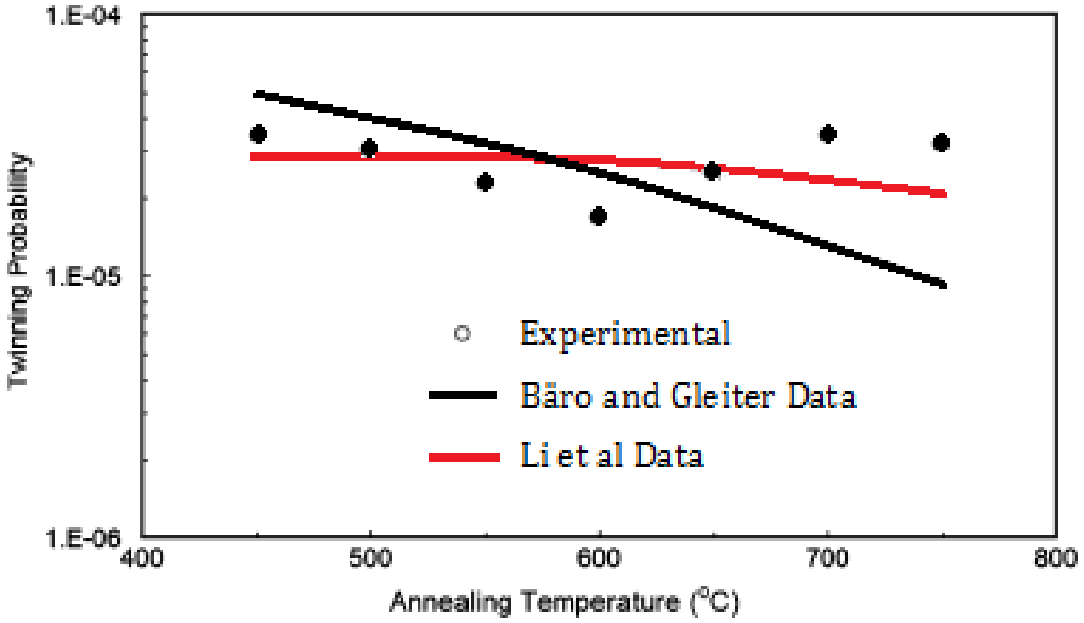


Figure 2.21 Comparison of calculated values for twin probability in Cu-3Wt.%Al with experimental values for a grain size of 300µm at various annealing temperatures [95].

Table 2.4 Values from Bäro and Gleiter [94] and Li et al [95] to calculate the twinning probability of Cu-3Wt.%Al for different annealing temperatures.

	Bäro and Gleiter [94]	Li et al [95]
Q	1.22×10^{-19}J/atom	1.22×10^{-19}J/atom
ε	0.238J/m^2	0.300J/m^{2*}
h	2.098×10^{-10}m	2.098×10^{-10}m
δ	0.476J/m^2	0.600J/m^{2†}
$\Delta G^{0\ddagger}$	7.596×10^{-26}J/atom	3.830×10^{-25}J/atom
$\sigma_z^{\S}$	0.00954J/m^2	0.01037J/m^2

* From Equation 2.28
† From Equation 2.27
‡ From Equation 2.30
§ The surface energy of a coherent twin boundary was an adjusted parameter to provide the best fit for the experimental data.

Varin [97] published the results from an investigation where the number of twins per grain was compared to grain size in types 316 and 316L stainless steel. The investigation concluded that the number of twins per grain is effectively constant up to 100μm and then varies linearly with grain size. The increase shown by Varin was small (one twin per grain increase over an increase of 300μm for grain size), and could be explained by the fact that not all twin boundaries are revealed by a single section; that is, a twin may exist in the grain volume below or above the polished section.

Dash and Brown [67] investigated the nucleation and growth of twins in a 78% Ni-Fe alloy using transmission electron microscopy. Two conclusions from the investigation supported those made by earlier researchers, i.e., twins nucleated at migrating grain boundaries, and the driving force for their formation is generated by a net reduction in surface energy. The theory proposed by Dash and Brown was that twins formed as a result of stacking fault packets at a migrating grain boundary driven by the process of dislocation annihilation.

Prior to 1978 all models describing the formation of annealing twins suggested there was a requirement for grain boundary migration. Meyers and Murr [98] proposed a model in which a twin 'pops out' of the grain boundary and grows into a grain via the migration of an incoherent interface. The driving force for the formation of an annealing twin is the overall reduction in dislocation density and interfacial energy by a two-stage process: initiation and propagation. Soon after the original model was proposed, Goodhew [99] confirmed the findings by Meyers and Murr by TEM analysis on thin gold bicrystals. This model is now referred to as "Grain Boundary Dissociation".

2.4.3 Σ3 Formation during Recrystallization

Because boundary plane indices are required to identify twins, studies often use Σ3 boundaries as a quasi-identification for a twin. Assuming an increase in Σ3 boundaries will result in an increase in twin boundaries, the studies can still provide valuable insight into twin formation during recrystallization.

Studies investigating the effect of recrystallization on the frequency of Σ3 boundaries were performed on brass [100] and Pb-Ca-Sn-Al alloy [101]. Both alloys were strained to a 30% reduction in thickness followed by heating. Results showed an increase in $\Sigma 3^n$ boundaries of 20% for brass and 40% for Pb-Ca-Sn-Al. The results demonstrate that recrystallization can produce different proportions of $\Sigma 3^n$ boundaries compared to the original microstructure.

Grovenor et al [102], examining the migration of HGBs, suggested that the formation of Σ3 boundaries aids the growth of a grain during recrystallization. The successful development and growth of a nucleated grain is partially based on the misorientation with the surrounding deformed matrix [29], i.e., some HGB misorientations promote faster migration of a boundary than others. This study [102] concluded that when a migrating grain boundary was seen to slow, the nucleation of a twin would result in an increase in boundary migration. This observation was also made by Jones [103], where studies on a 20wt.%Cr 25wt.%Ni steel and alpha brass showed Σ3 boundaries formed at grain boundaries during the early stages of recrystallization.

Field et al [61], with the assistance of EBSD mapping and a heating stage, were able to perform in-situ investigations on the recrystallization of copper, essentially producing a snapshot in time of the microstructure during the annealing process. It was shown with EBSD mapping that when an interface of the growing grain slowed, a Σ3 would form and the boundary velocity would increase. The boundary velocity was calculated from the increase in grain radius as a function of time. A boundary typically slowed when it grew into a region of the deformed matrix with low dislocation density or a misorientation similar to that of the recrystallizing grain. The investigators concluded that the growth during recrystallization is dependent on Σ3 boundaries forming. Since the driving force for grain growth is a thermally activate diffusion based process, Σ3 formation may be required to provide the necessary misorientation between the grain and the deformed region at low temperatures. At higher temperatures, the growth of grains is less dependent on Σ3 formation, since an increase in diffusion occurs due to thermal energy. The number of twins per grain decreased from 6.4 to 3.2 with an increase in annealing temperature from 155°C to 400°C.

As with temperature, the amount of deformation in a material prior to annealing can affect the recrystallization rate. Gerber [104] presented results suggesting that Σ3 formation during recrystallization for copper was greater for the sample with 70% strain in comparison to the sample with 90% strain. A similar result was identified in a study of cold-drawn copper wires [105], in which the wire strained to 52% reduction in area produced a Σ3 length fraction 18% higher than the sample strained to 94%. The results suggest that the increase in stored energy as a result of higher deformations provides the driving force for recrystallization and therefore there is less dependence on Σ3 formation.

Kumar et al [106] examined the effect of recrystallization on the length fraction of Σ3 boundaries in 304 stainless steel. Samples were cold rolled to either 60 or 80% reduction in thickness followed by annealing between 700°C and 1000°C for one hour. Results showed the sample strained to 80% produced a Σ3 length fraction 22% less when compared to the sample strained at 60%.

2.4.4 Grain Boundary Engineering (GBE)

Grain Boundary Engineering (GBE) was originally proposed by Watanabe [107], under the term 'grain boundary design', whereby an increase in the relative fraction of low CSL boundaries (typically $\Sigma 3^n$, where n = 1-3) results in the break-up of the network of random HGB. The implications of the reduction in random HGB connectivity are that damage mechanisms dependent on the network, such as intergranular stress corrosion, grain boundary sliding, or Coble creep, will be suppressed. EBSD is typically used to identify the low CSL boundaries in GBE studies.

The mechanisms for the development of GBE microstructures are usually explained in terms of the "prolific twinning" [108] of materials with low stacking fault energies, in particular, FCC materials. Processes developed with the primary purpose of increasing the proportion of $\Sigma 3^n$ boundaries rely on combinations of strain and annealing.

Kumar [109] noted that recrystallization would be undesirable in GBE because the grain boundary network is effectively replaced rather than modified. A modified grain boundary network is more likely to disrupt the connectivity of random grain boundaries. Therefore, the formation of GBE microstructures is performed by strain-annealing, a process whereby a balance between ensuring a suitable amount of strain energy is supplied to promote the migration of existing boundaries and the formation of $\Sigma 3$ boundaries, but not so much as to create an entire new microstructure through recrystallization, is obtained.

Lee and co-workers discussed the effect of combinations of low tensile strain (3% to 12%) and annealing (500°C to 900°C for 10 minutes) on the CSL boundary ($3 \leq \Sigma \leq 29$) length fraction in nickel [110]. When temperature was less than 800°C, the grain size remained constant and the material showed an increase in hardness compared to the as-received sheet. At 800°C, all samples displayed an increase in grain size and a decrease in hardness compared to the as-received sheet. No significant increase in CSL boundary fraction was observed. Only at 900°C did an increase in CSL boundary fraction occur. This increase in CSL boundary fraction was largest at the intermediate strain level of 6%, and was accompanied by a two times increase in grain size. In a later publication [111] it was shown that given sufficient time (30 minutes) at 800°C all the nickel samples, regardless of strain (3% to 10%), eventually achieved the same proportion of CSL boundaries. Similar results to those presented by Richards are seen in other investigations. Thomson and Randle [59, 112] showed that for nickel, a compressive strain of 6% followed by 24 hours at 750°C significantly increased the $\Sigma 3$ boundary fraction.

Iterative thermo-mechanical processing (multiple strain-anneal cycles) is known to produce a microstructure with a higher frequency of boundaries with low CSL ($3 \leq \Sigma \leq 29$) values. Li et al [113] showed the effect of iterative treatments on the CSL boundary length fraction of Inconel 718. Three compressive strain values, 2.5%, 5% and 7.5%, and 10 minutes at 1020°C were repeated three to five times. However, after some iterations the hardness was greater than the previous step, suggesting that the temperature or time was insufficient for boundary migration to take place. This lack of grain boundary migration resulted in CSL boundary fractions similar to those observed in the preceding iteration. With the material still in a strained state, additional energy supplied by the next iteration provided the driving force for grain boundary migration, resulting in an increase in CSL boundary fraction ($3 \leq \Sigma \leq 29$). A similar observation was made by Randle [114] in the iterative processing of copper using strains of 6% and five minute anneals at 665°C.

A study by Drabble [115] on Alloy 800H suggests that for low strain levels, a longer annealing time at some intermediate temperature increased the special boundary fraction most significantly. Drabble also showed that for longer annealing times the retained strain, identified through hardness testing after each iteration, was less than for shorter times.

Randle provides an explanation for the formation of Σ3 boundaries with the Σ3 Regeneration Model [116] illustrated schematically in Figure 2.22. Figure 2.22(a-b) shows the migrating boundary of the left hand grain impinging on the right hand grain. Figure 2.22(c-d) shows that with further migration the Σ3 in the left had grain will intersect with the Σ3 in the right hand grain and a Σ9 will form. Figure 2.22(e-f) indicates that the Σ9 will migrate towards another Σ3 causing the boundaries to intersect and form a new Σ3 boundary. The model relies on the condition that an encounter between a Σ9 and Σ3 or a Σ9 and Σ27 will generate a Σ3 boundary on the basis of lower energy compared to higher $\Sigma 3^n$ boundaries.

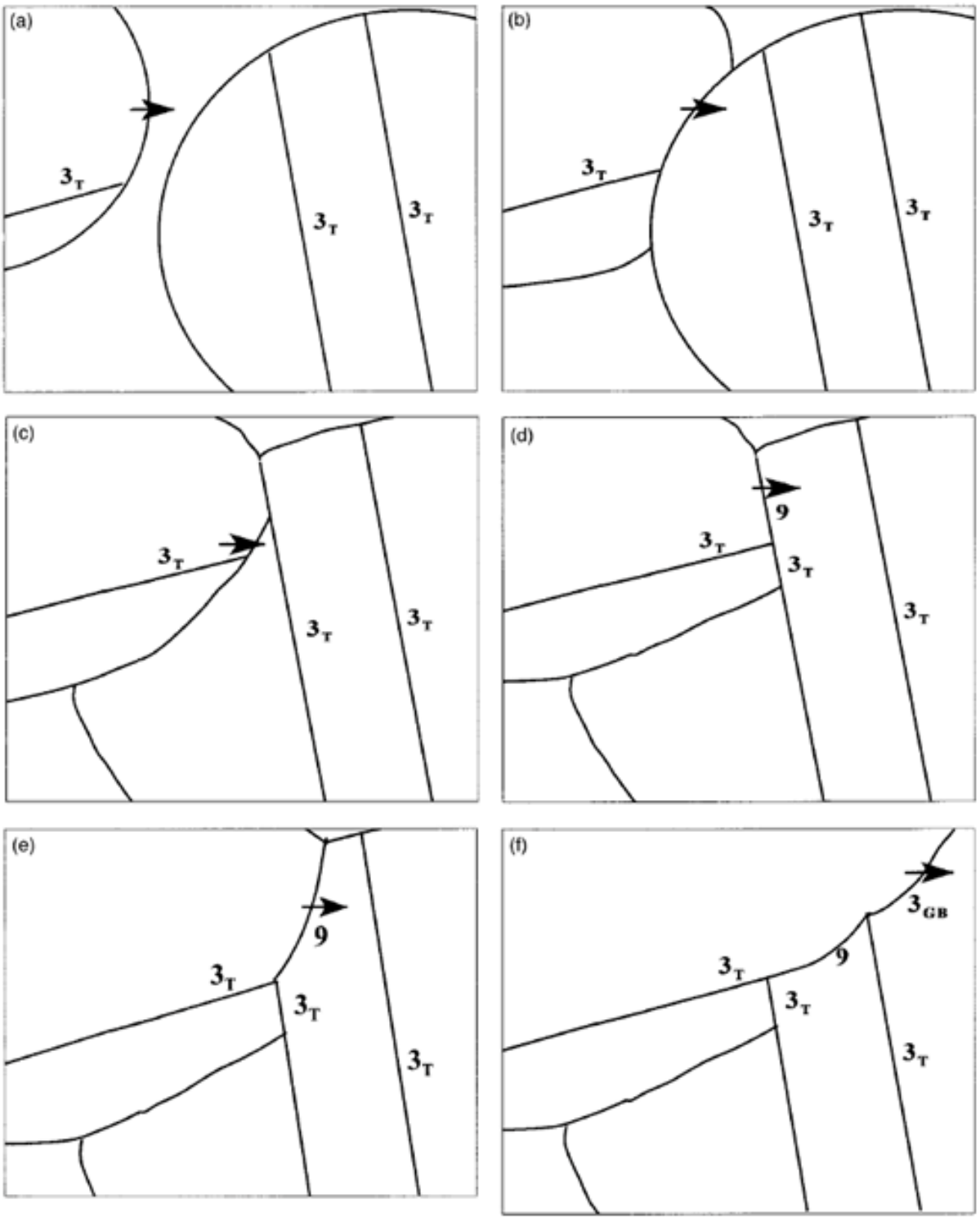


Figure 2.22 Schematic of twin interactions and the Σ3 generation model [116].

(a-b) Migrating grain boundary impinging onto right hand grain.

(c-d) Σ3 twin encounters another Σ3 twin resulting in the formation of a Σ9.

(e-f) Σ9 continues to migrate encountering another Σ3 resulting in the formation of a non-coherent, mobile Σ3 grain boundary.

Research performed by Reed and Kumar [62, 117] casts doubt on the validity of the Σ3 regeneration model. Reed [117] outlines a general approach to consider the inter-relationships among the various angle/axis pairs ($\omega/\hat{n}$) describing misorientations between grains producing CSL boundaries. An example is shown in Figure 2.23 for the $\Sigma 3^n$ CSL, described by the authors as the twin-related group, 60°/<111>. Each node represents a boundary misorientation, with the four paths (a, b, c, and d) representing the four fundamental Σ3 operations in a cubic crystal, that is, a 60° rotation about one of the four <111> axes.

Using the mathematical model, Reed et al [62] analysed the "Σ3 regeneration model" [116]. The model (Figure 2.22) starts with two separated twin-related domains (TRDs) and ends with a structure in which the TRDs have grown together leading to the formation of several new boundaries. While the model assumes the nature of the boundaries formed, the process used by Reed suggests statistically different outcomes are more probable than the formation of a Σ3. When the Σ9 boundary intersects with the Σ3 boundary, Randle suggests that the outcome from this "reaction" will be the formation of a Σ3, although the model suggests only a one in four chance for the formation of the Σ3. The one in four chance is illustrated by first identifying a Σ9 node in Figure 2.23, and then recognising only one axis rotation would result in the formation of a Σ3, while the other three rotations result in the formation of Σ27 boundaries.

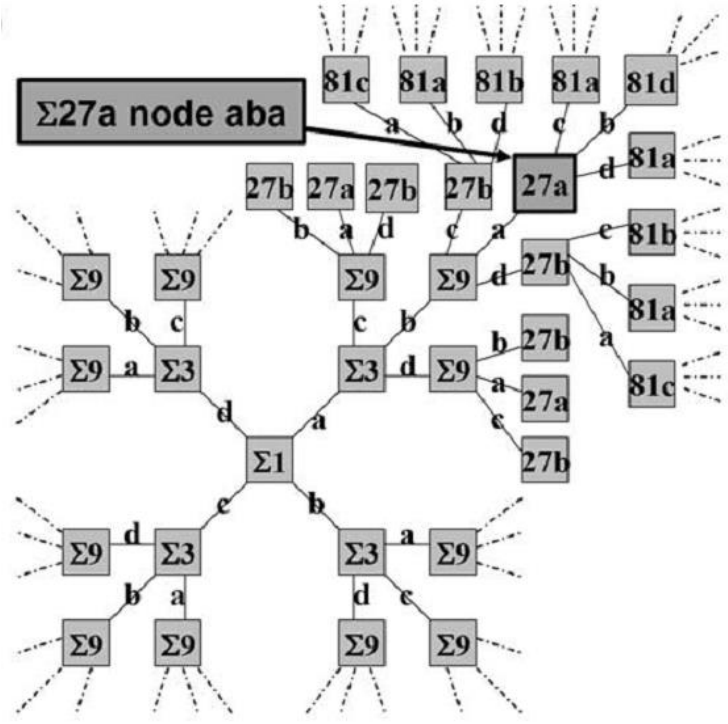


Figure 2.23 Map showing the 60°/<111> required to obtain CSL boundary types pertaining to the $\Sigma 3^n$ group of boundaries [62].

Reed concludes that the "Σ3 regeneration model" carries the hidden implication that the regions A and B (see Figure 2.24) have exactly the same orientation, and the odds of this happening by chance are negligible. While the "Σ3 regeneration model" describes a mechanism is which grain boundary connectivity is disrupted by the interactions and resulting formation of $\Sigma 3^n$ boundaries, it lacks a convincing argument as to why the boundary formed should be a Σ3. Randle suggests [108] the formation of a Σ3 is supported because when Σ27 populations are measured in a grain boundary network values are typically low. It is important to remember these statistics are measured in 2D, and Reed et al [1] shows that when measured in three-dimensions, the Σ3 number fraction is 33% less than that measured in 2D, and the Σ27 number fraction is approximately 20% higher. Overall, the results from Reed [1] suggest GBE processing will increase the number fraction of Σ3 boundaries, but not to the same extent as one would be led to believe from the "Σ3 regeneration model" as it is presented in Figure 2.22.

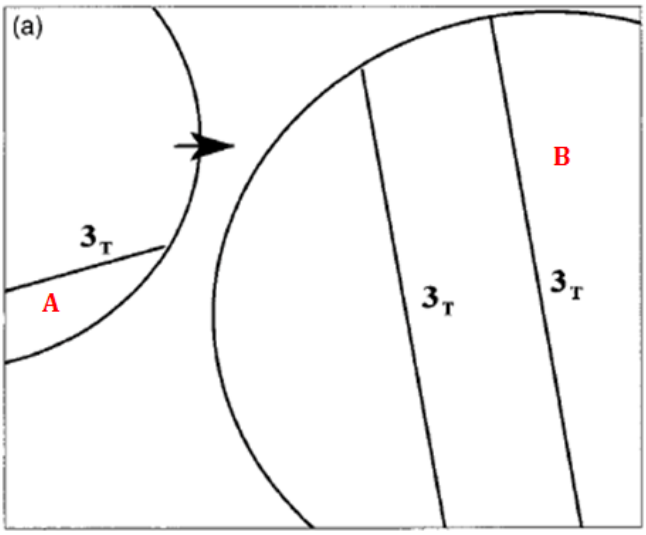


Figure 2.24 "Σ3 regeneration model" showing grain orientations A and B in two separate TRDs. For the "Σ3 regeneration model" to hold as proposed, A and B will have to have the same orientation.

While the effectiveness of GBE processing has typically been measured by the resulting $\Sigma 3^n$ population densities, the break-up of the random high angle grain boundary (RHGB) network is also seen to have significant implications on mechanical performance [118]. Several methods exist for the quantification of network connectivity including the Neutral-Twin Concept [60], Triple Junction Geometry [119], and the Effective Resistance Model [115].

The neutral twin concept proposed by Lehockey et al [60] attempts to divide Σ3 boundaries into two groups: those which disrupt the RHGB network, and those which did not. The calculation of special

boundary fraction excluded the Σ3 boundaries which did not disrupt the grain boundary network. These boundaries were termed “neutral twins”.

The analysis of triple junctions in a GBE material was proposed by Kumar et al [119], and involves assigning each triple junction a category based on the number of $\Sigma 3^n$ ($n \leq 3$) boundaries which meet at that junction. Triple junctions are identified from EBSD maps, and each grain boundary forming a triple junction defined as either non-$\Sigma 3^n$ or $\Sigma 3^n$. The triple junctions are categorised depending on whether they contained 0, 1, 2, or 3 $\Sigma 3^n$ boundaries. The authors showed that as a result of GBE, the relative fraction of triple junction containing zero $\Sigma 3^n$ boundaries is decreased, and that the fraction containing three $\Sigma 3^n$ boundaries is increased.

Drabble et al [115] introduced the effective resistance model which describes a grain boundary network analogous to an electric circuit by representing grain boundaries as resistors. The resistance of the grain boundaries varied depending on boundary length and character. The model assumes that current flow though a circuit is comparable to the diffusional flow though a grain boundary network, and therefore correlations may be made with grain boundary diffusion-based material properties, such as Coble creep rate.

Drabble et al [115] applied the effective resistance model to Alloy 800H grain boundary engineered material. Samples with high $\Sigma 3^n$ length fractions and disrupted RHGB networks showed an increase in resistance compared to the as-received material. The triple junction geometry was analysed with results showing samples with highest resistance also had the highest fraction of junctions joining two and three $\Sigma 3^n$ boundaries. Drabble concluded that there was a reasonable correlation between the triple junction geometry model and the effective resistance model.

2.5 Summary

In summary, the average grain size and twin populations are controlled by the nucleation and growth rates of grains during recrystallization. An increase in the nucleation rate produces a finer grained sample, while an increase in the growth rate produces coarser grains. Both the nucleation and growth rates increase with increased deformation and diffusion activated by an increase in temperature. The increase in deformation provides stored energy as a driving force for growth, but more notably the dislocation boundaries provide ideal sites for nucleation. Annealing temperature activates diffusion processes allowing for the nucleation of grains, but more so for growth.

The formation of twin boundaries is also affected by the recrystallization rate. Several studies [61, 102, 104, 106] showed that when a high-angle grain boundary slowed, the formation of a Σ3 boundary provided the necessary driving force for growth to accelerate. Field [61] showed the formation of Σ3 boundaries through high speed EBSD mapping of in-situ recrystallization of copper. The EBSD maps were assumed to be 'snap shots' in time and were used to measure grain boundary velocity. When a boundary migrated into a deformed region of low dislocation density, or had a similar orientation with that of the recrystallizing grain, the formation of a Σ3 produced the necessary conditions to accelerating growth.

Through the strain-annealing technique, the population of Σ3 boundaries (of which coherent twins are a subset) can be increased. The strain-annealing technique involves using low strains to provide a driving force for the migration of existing grain boundaries and the formation of Σ3 boundaries. The Σ3 regeneration model [116] describes the formation of additional $\Sigma 3^n$ boundaries though boundary interactions. While Randle [116] suggests that the majority of these new boundaries would be Σ3s, models produced by Reed and Kumar [62] suggest the formation of Σ9s and Σ27s are more probable.

While many studies have been published on recrystallization rate, grain size, and the formation of Σ3 boundaries, few have focused on Alloy 800H specifically. This thesis reports on a study performed on Alloy 800H where the degree of cold work and annealing temperature was varied between samples to better understand the final grain size statistics and grain boundary character of recrystallized samples. The information provided from this study will assist in the processing of Alloy 800H in industrial environments and future Alloy 800H research studies.

Finally, the morphology of twin boundaries was discussed. In 2D, twin boundaries were described as one sided twins, complete parallel-sided twins, incomplete parallel-sided twins, and island twins [64]. In 3D [73], it was observed that twin morphology could be simply described as lamella twins and edge twins.

Lamella twins are characterised by the existence of two parallel twinning planes and a section taken through a lamella twin volume can produce any of the four 2D morphologies. An edge twin only had one twin interface so that sectioning would always produce the 2D morphology described as a one-sided twin.

While the literature details numerous studies of twinned alloy systems examined in 3D [1, 73], few investigations were designed specifically to understand and document the morphologies and crystallography of twin boundaries. This study provides an ideal opportunity to employ EBSD strategies to further understand the crystallographic and morphological character of twin boundaries and make a valuable contribution to the literature.

Studies [82-84] measuring the boundary plane indices of Σ3 boundaries concluded that a straight boundary trace appearing on a planar section does not necessarily suggest the presence of a coherent twin. This result implies that the identification of coherent twins from misorientation measurement and/or 2D morphology is not sufficient. Trace analysis was discussed as a method of identifying a potential coherent twin, without the laborious task of serial sectioning.

Chapter 3 Experimental Methodologies

3.1 Introduction

Chapter 3 details the material preparation techniques and methodologies used to identify coherent twins and obtain grain size measurements. In general, the metallographic preparation procedures used for austenitic stainless steels are fairly well established and have been detailed elsewhere (e.g. [120]). This chapter will focus principally on the processes involved in the effective collection of electron backscatter diffraction (EBSD) data and the post-processing required to analyse the results.

This chapter includes four main sections:

1. The thermo-mechanical procedures employed to generate microstructures.
2. The preparation methods for producing samples suitable for analysis.
3. Optical and EBSD techniques for analysing the microstructure of Alloy 800H.
4. Analytical methods developed to extract pertinent information from EBSD data.

3.2 Thermo-Mechanical Processing

3.2.1 Cold Rolling and Annealing

Deformation of the material was achieved by cold-rolling. The total reduction in thickness (RT) was achieved through the accumulation of several passes with a reduction of approximately 0.25mm per pass. The RT was measured at each pass using digital callipers, and the final reduction was within 0.05mm (approximately 0.5%) of the target thickness. Samples with a RT between 6% and 80% were produced in this study.

Annealing of the material was performed in a tube furnace. Furnace temperature was measured by an N-type thermocouple positioned inside the furnace cavity next to the sample. An accuracy of ±5˚C was observed. The variation in temperature was caused primarily by the drop in temperature experienced during sample exchange. Heating time was controlled to ±5 seconds. The samples were water-quenched after heating to prevent the formation of precipitates during cool down. Temperatures between 1000°C and 1350°C were used in this study. The temperature range was selected to ensure comparable conditions to those typically employed in manufacturing. Special Metals [2] documentation indicates that for annealing temperature less than 1000°C for samples with 20% strain, a minimum of 1 hour

would be required for the onset of recrystallization. For temperatures greater than 1350°C there is a risk of localised melting.

3.2.2 Sample Preparation

Plate

Metallographic samples were cut from strips with the transverse section presented for analysis. The transverse section is shown in Figure 3.1 relative to the sample strip. Initially, both the transverse and longitudinal sections were analysed; however, it was found that there was a negligible difference in grain size and shape between the two sections. To limit any effects caused by cold-rolling, the prepared surface was at least 20mm away from the end of the strip. The samples were hot mounted in a Buehler ProbeMet copper filled conductive mounting compound.

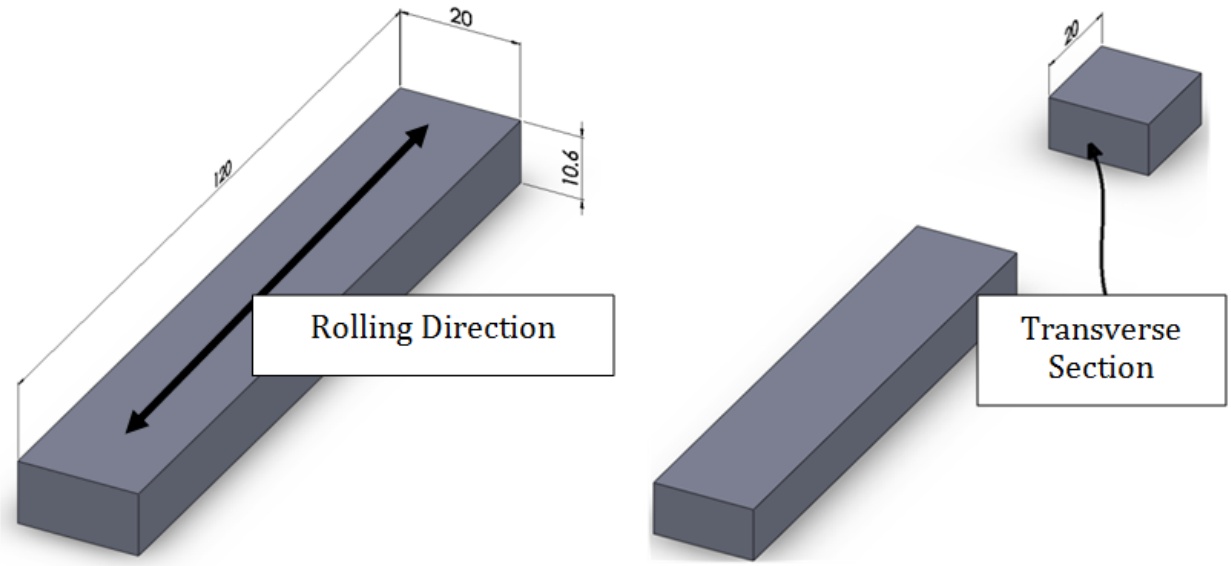


Figure 3.1 Sample strip illustrating rolling direction and surface of interest.

Pipe

Metallographic samples prepared from the pipe material were cold-rolled and annealed using similar methods as for the plate samples. Rings of 15mm thickness were cut from the pipe length and then cold-rolled to reduce the thickness. Metallographic samples were cut from the rings and mounted so that the transverse section was analysed, Figure 3.2. Initially, both the transverse and radial sections

were analysed; however, just as in the case of plate samples, no measureable difference was observed between the two sections.

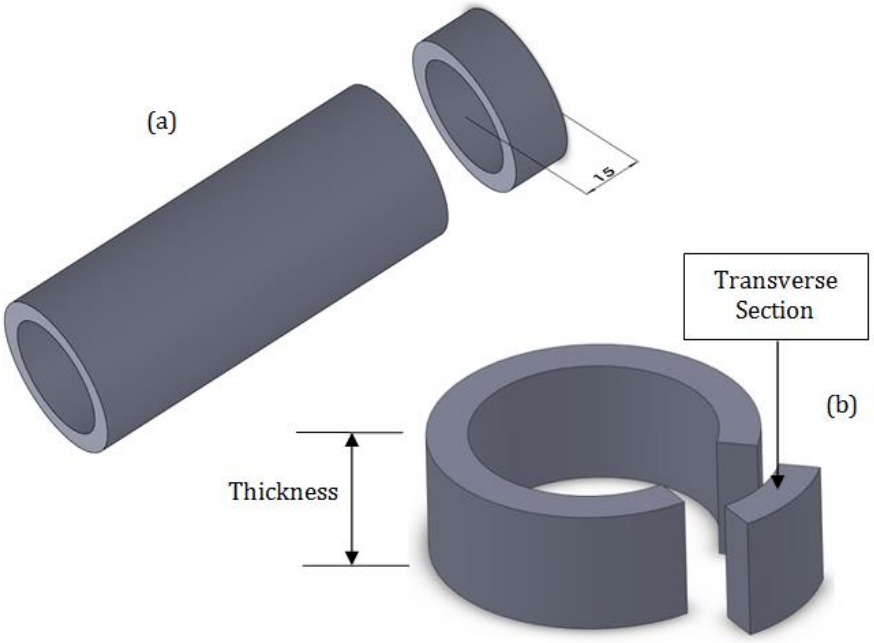


Figure 3.2 Sample sectioning procedure for pipe material: (a) ring cut from straight pipe length, (b) metallographic sample cut from ring.

Each sample was ground to a 600-grit finish in a five step process (180, 240, 320, 400, and 600-grit) using silicon carbide (SiC) paper. The samples were subsequently polished in three steps (9µm, 3µm, and 1µm) using Buehler MetaDi diamond suspensions. Final polishing was performed by a Buehler MiniMet Automatic Polisher, using a solution of 0.06µm colloidal silica suspension (Buehler MasterMet). Typical polishing times were in the range of 60-120 minutes.

For EBSD analysis, the final polish was performed immediately before placing the sample in the scanning electron microscope (SEM) to minimize surface contamination. Figure 3.3 shows a comparison between EBSD patterns taken from a well prepared sample and a poorly prepared sample. The mean angular deviation (MAD), the goodness of fit between the simulated diffraction pattern and the captured electron backscatter pattern (EBSP), for the well prepared sample equalled 0.308° while the poorly prepared sample produced a MAD of 1.12°. Any diffraction pattern that produced an MAD greater than 1° was unindexed. A map of sufficient quality would produce an average MAD of 0.5° or less, and contain no more than 10% unindexed pixels [121].

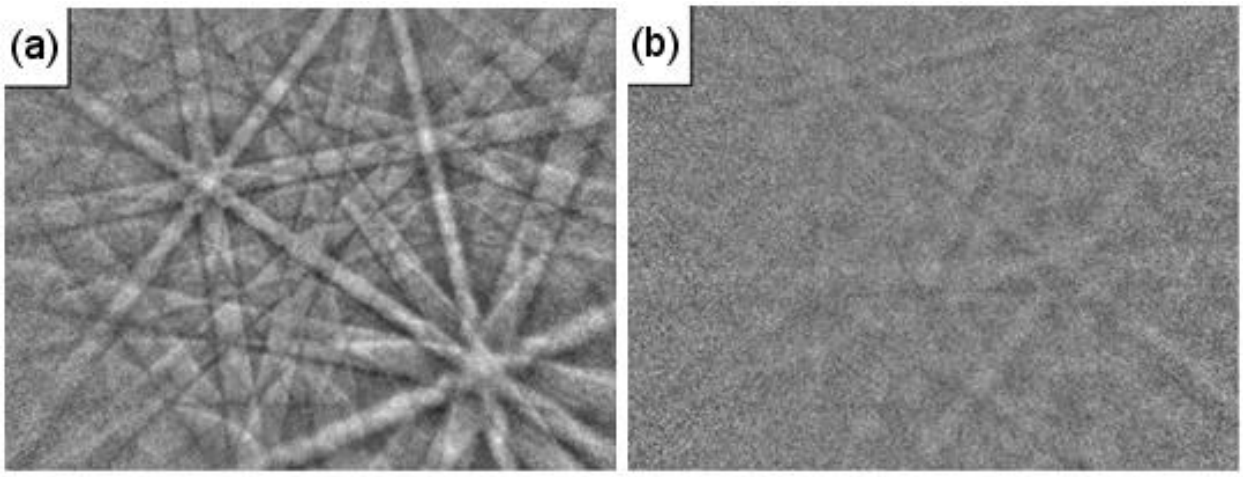


Figure 3.3 Comparison between EBSD patterns taken from a well prepared sample (a), and a poorly prepared sample (b).

3.2.3 As-Received Material

The as-received (AR) Alloy 800H was obtained from two sources. The processed pigtail pipe lengths were obtained from Methanex New Zealand and the plate material was obtained from ThyssenKrupp VDM Australia Pty. Ltd.

Table 3.1 summarises the average grain sizes (measured from EBSD maps) for the AR materials. The values indicate the grains are roughly equiaxed, a result consistent with a fully recrystallized material. The EBSD maps of the AR plate material, Figure 3.4, show a homogenous microstructure with no obvious elongation of grains. This is consistent with the grain size measurements, Table 3.1.

Table 3.1 Grain size measured from EBSD maps for the As-Received Alloy 800H.

As-Received	Grain Size (EBSD)
Plate – Longitudinal	65.2 µm ± 10.2%
Plate – Transverse	66.2 µm ± 8.6%
Pipe – Transverse	49.3 µm ± 9.2%
Pipe – Radial	52.1 µm ± 9.8%

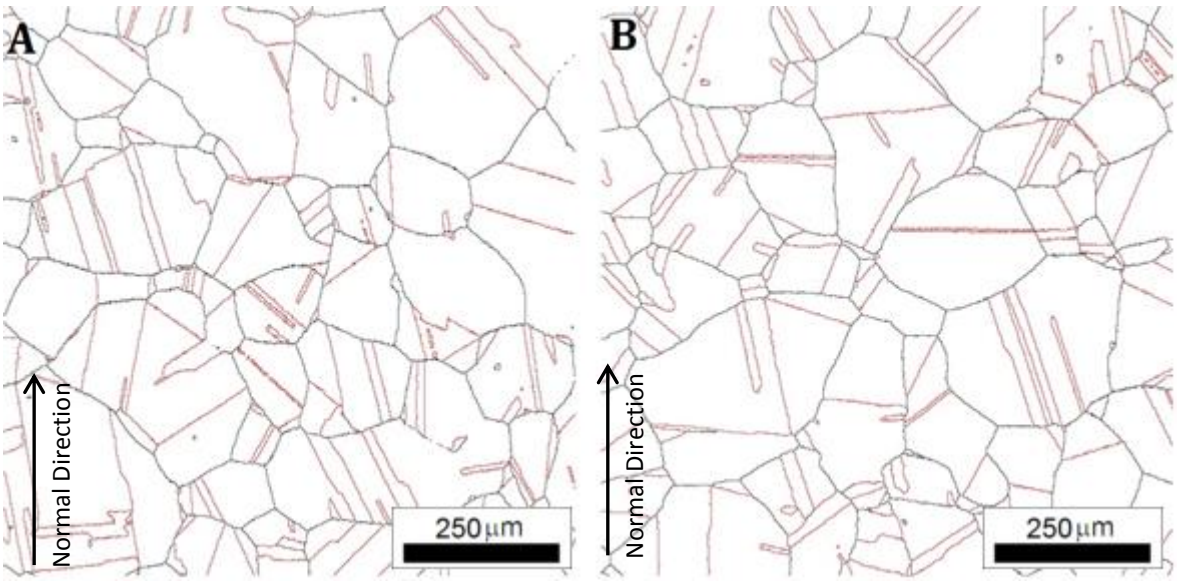


Figure 3.4 EBSD maps of the as-received material (plate) sectioned along longitudinal (a), and transverse (b) directions. Σ3 boundaries identified as red, all other high-angle grain boundaries (HGB) in black.

The grain orientation spread (GOS) [122] can be used to determine the recrystallization fraction. GOS calculates the mean misorientation between the average orientation of the grain and the orientation of individual pixels (EBSD measurements) that form the grain. The GOS is given as:

$$GOS = \frac{1}{N}\sum_{P=1}^{N}\left\{min\left[\cos^{-1}\left(\frac{trace[g_{ave}(h_i g^P)^{-1}]-1}{2}\right)\right]\right\}$$

Equation 3.1

where g_{ave} is the orientation matrix for the average orientation of the grain, h refers to a symmetry matrix, and g^P is the orientation matrix for a given data pixel in the grain.

Figure 3.5, shows an EBSD map from the AR plate with grains coloured to represent the orientation spread in each grain. For comparison, the grain orientation spread was also calculated for a partially recrystallized specimen, Figure 3.6. Grains with a GOS value of less than 4° were considered recrystallized, whereas those greater than 4° were considered deformed [123]. Samples with more than 95% of the total area recrystallized are considered to be fully recrystallized. The AR and partially recrystallized specimens had recrystallization fractions of 0.95 and 0.55 respectively.

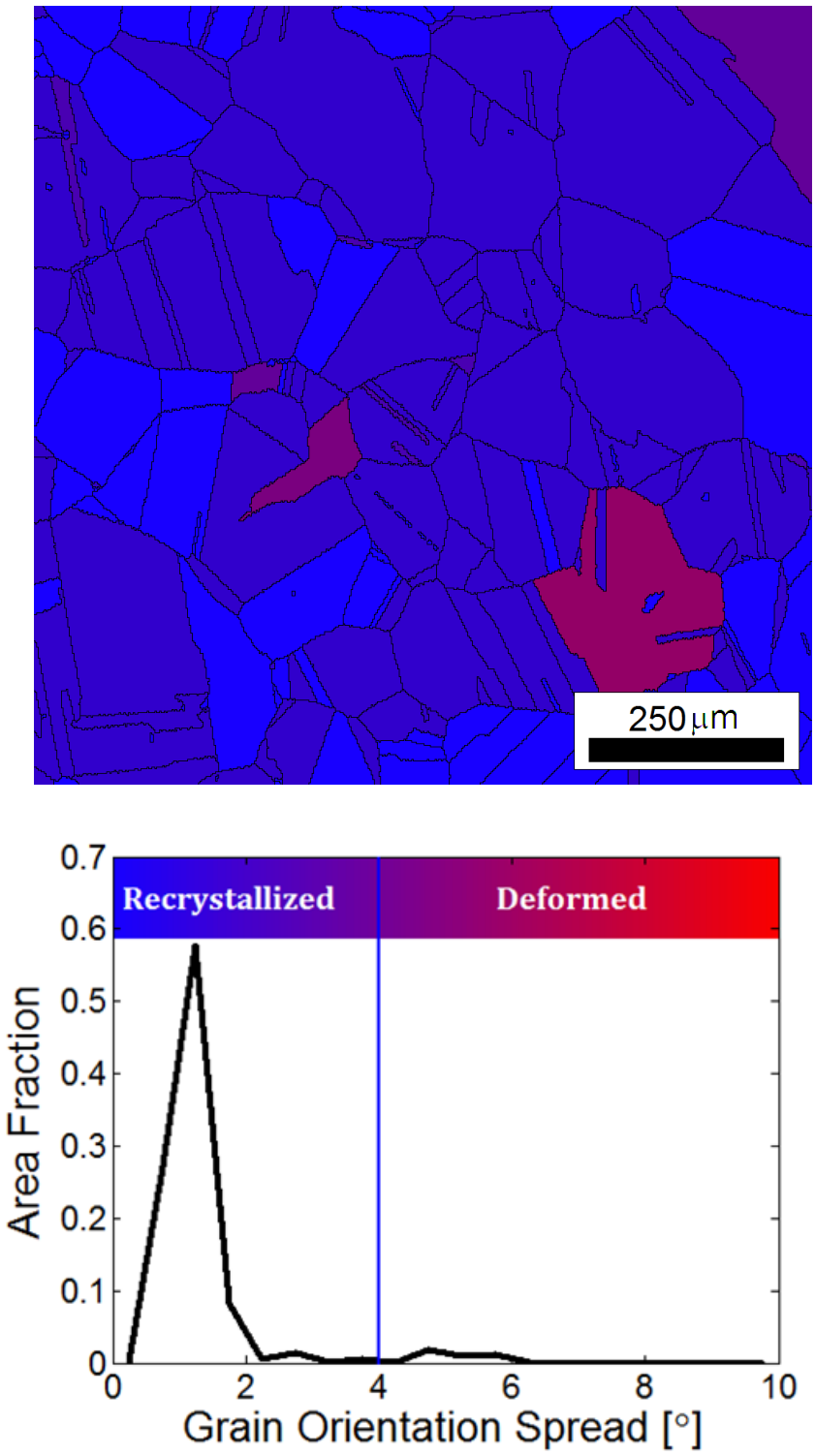


Figure 3.5 EBSD map indicating the grains defined as recrystallized (blue) or deformed (red) for the As-Received sample.

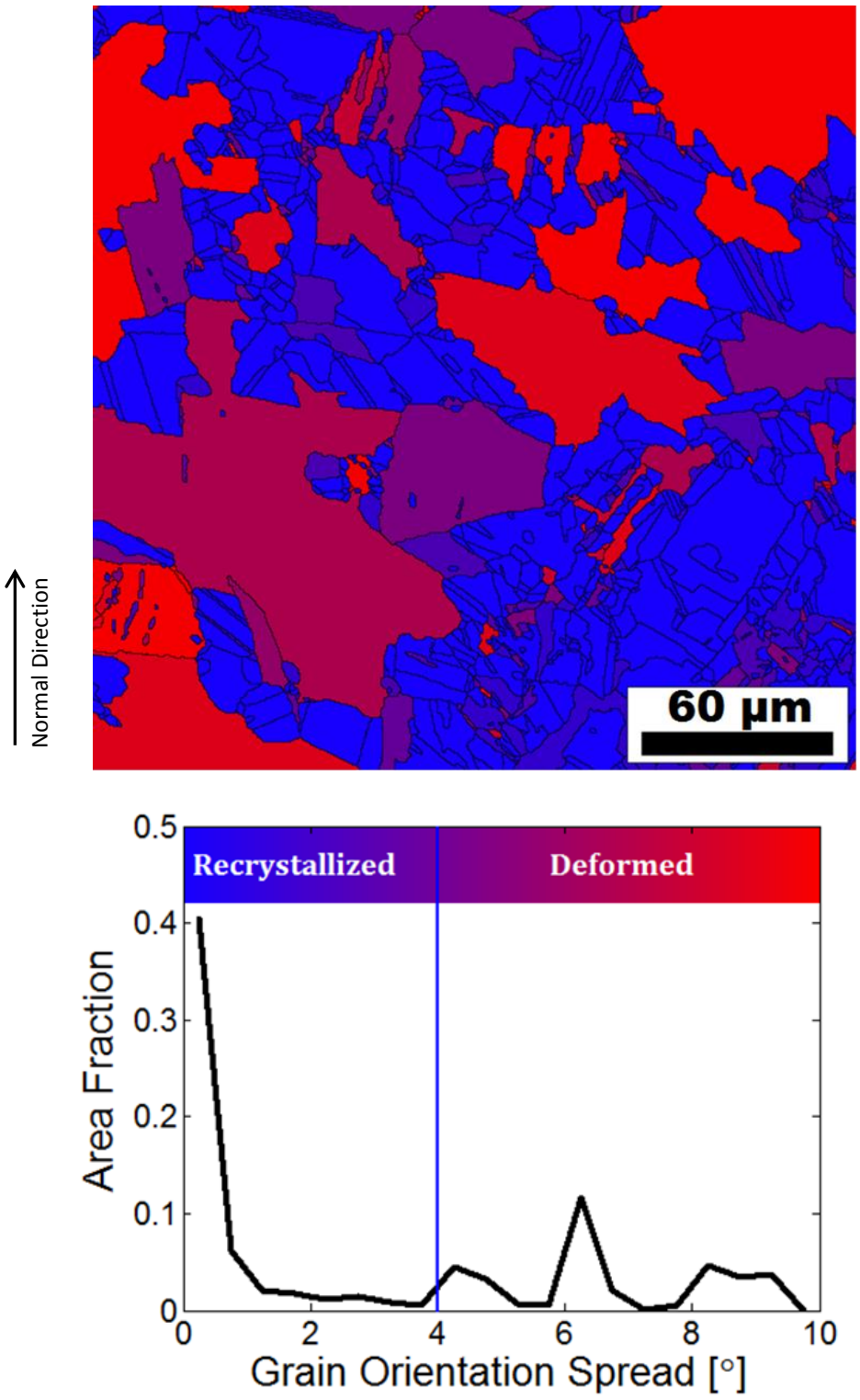


Figure 3.6 EBSD map indicating the grains defined as recrystallized (blue) or deformed (red) for the partially recrystallized sample.

3.2.4 Sample Texture

In order to compare different sample conditions in creep tests, it was necessary to determine if thermo-mechanical processing affects the microtexture of the samples. Rolling and annealing may induce an annealing texture, or certain preferred crystallographic orientations. It is also known that texture can have an effect on creep rates, although this is thought to occur predominantly at high strain rates [124].

Figure 3.7 compares the microtextures of three samples; Alloy 800H as-received plate, Alloy 800H plate cold worked to 20% than annealed for 30minutes at 1200°C, and post service Alloy 800H pigtail sample. The microtexture was calculated using the HKL software and the results are presented as inverse pole figure in Figure 3.7. For the plate material, the rolling direction corresponds with the z direction shown, and for the pigtail pipe the z direction corresponds to the pipe length.

The as-received plate had a maximum multiple of uniform density (MUD) of approximately four, the highest of the three samples analysed indicating slight texture. Poles at <001>$||_Z$ and close to <101>$||_X$ indicates a possible Goss orientation which is typically found in austenitic steels [125] and nickel [58]. However, with a maximum MUD of less than four, the as-received plate texture can be classified as weak.

After cold rolling and annealing the texture becomes even less uniform showing a MUD of less than three. Multiple twinning operations, such as those occurring during recrystallization, have been shown to weaken the texture [126]. The pigtail material displayed a slightly lower MUD (approximately two) than that of the processed plate indicating an essentially random texture.

Neither of the three samples showed any significant texture, and therefore the effect of microtexture was considered negligible in this work.

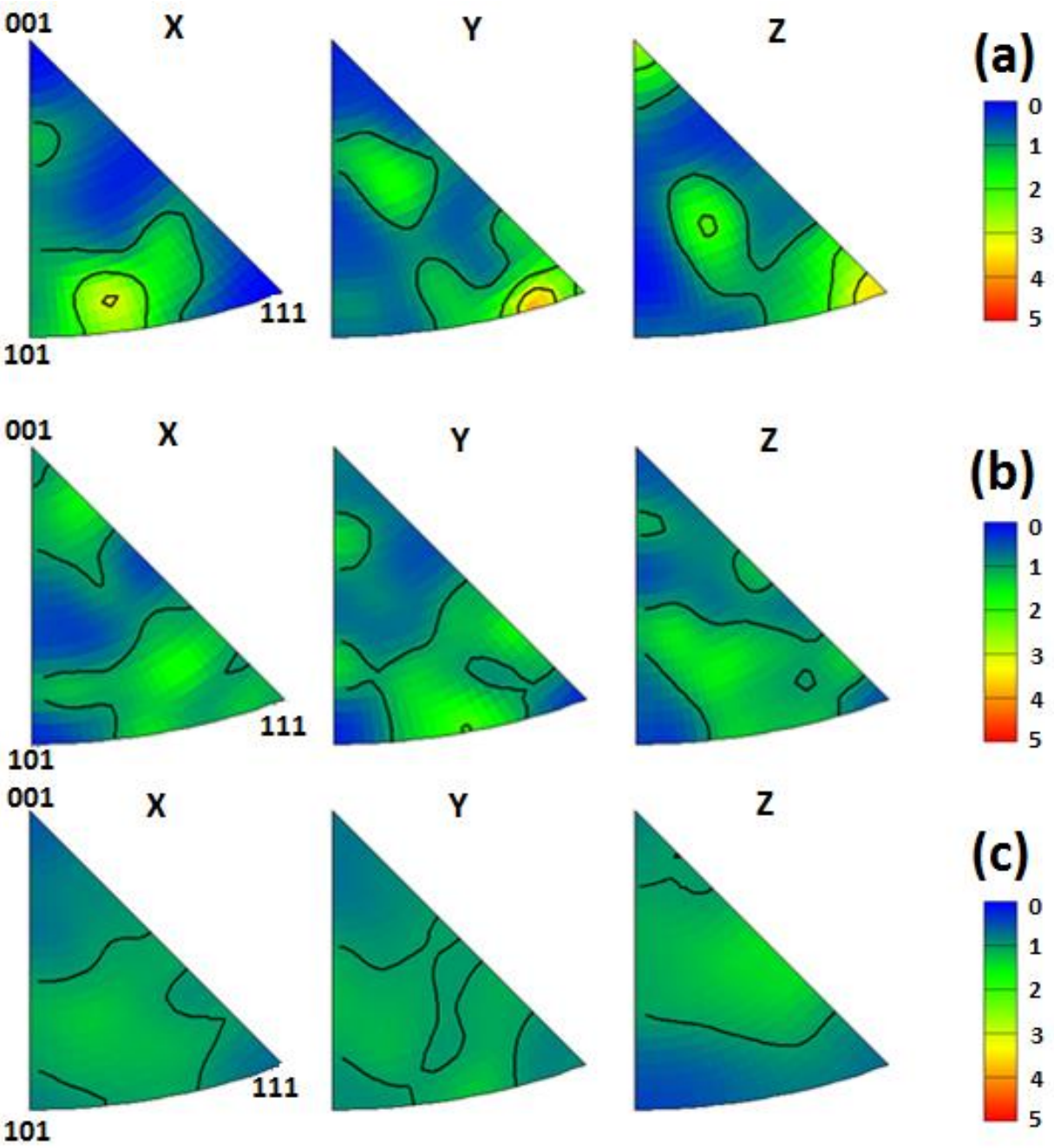


Figure 3.7 Comparison of microtexture between (a) as-received plate, (b) cold-worked and annealed plate and (c) post service pigtail.

3.3 Optical Micrographs

Etching was performed using glyceregia (30 ml glycerol, 40 ml HCl, 10 ml HNO_3) for 3-4 minutes to reveal grain boundaries. A typical result from the etching procedure is shown in Figure 3.8. In this study, a Leica DM-IRM inverted optical microscope coupled with a Nikon Digital Sight DS-Fi1 camera was used. Optical micrographs with a resolution of 0.07 µm/pixel at 500x magnification were possible with the described camera setup.

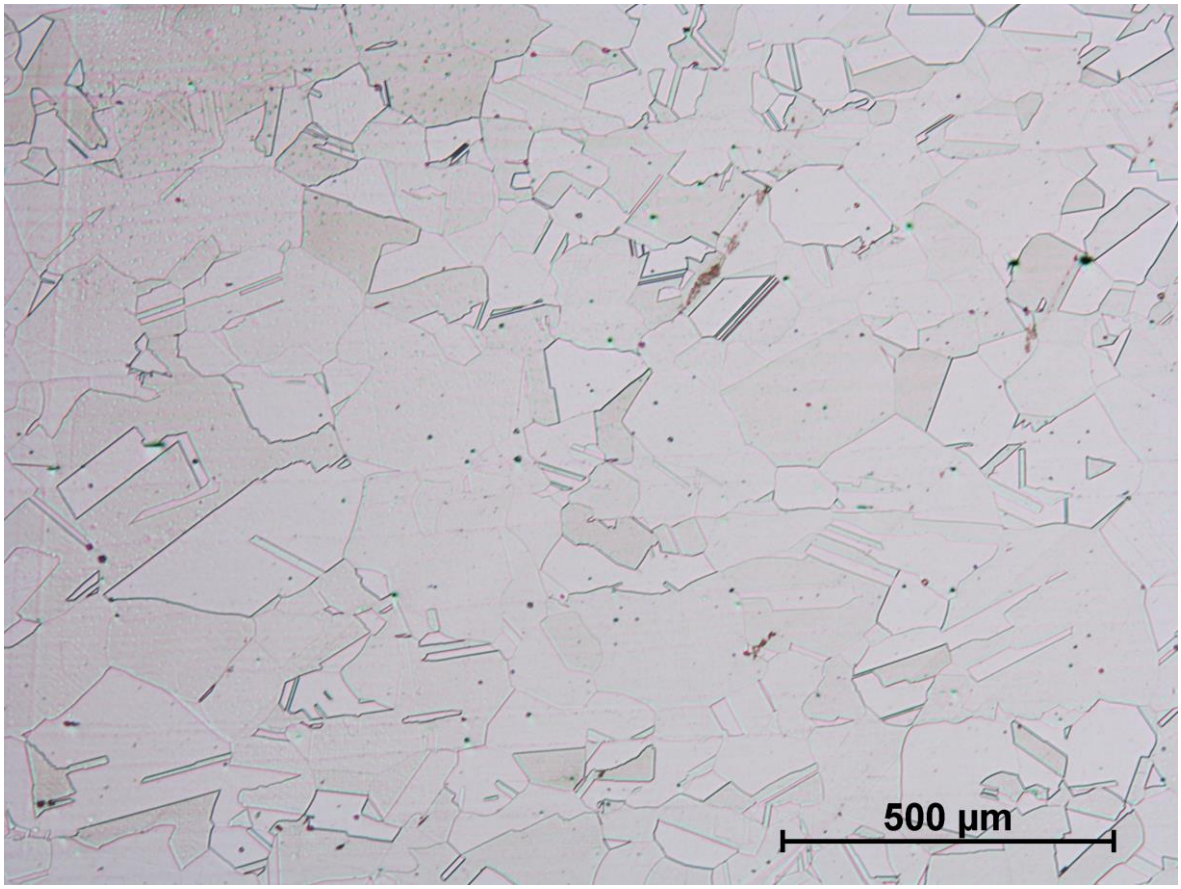


Figure 3.8 Optical micrograph showing the typical etch quality of Alloy 800H.

A limitation of optical microscopy is that not all boundaries are revealed through etching. For Alloy 800H, it was often observed that some boundaries, particularly Σ3s, later identified through EBSD mapping, were absent from optical images. Figure 3.9 shows four boundaries identified as Σ3 from EBSD mapping absent from the optical image. Conversely, Figure 3.9 indicates (blue circle) a situation where the EBSD mapping resolution was insufficient and failed to identify a pair of parallel Σ3 boundaries. The effect of EBSD mapping resolution on the identification of boundaries is discussed next.

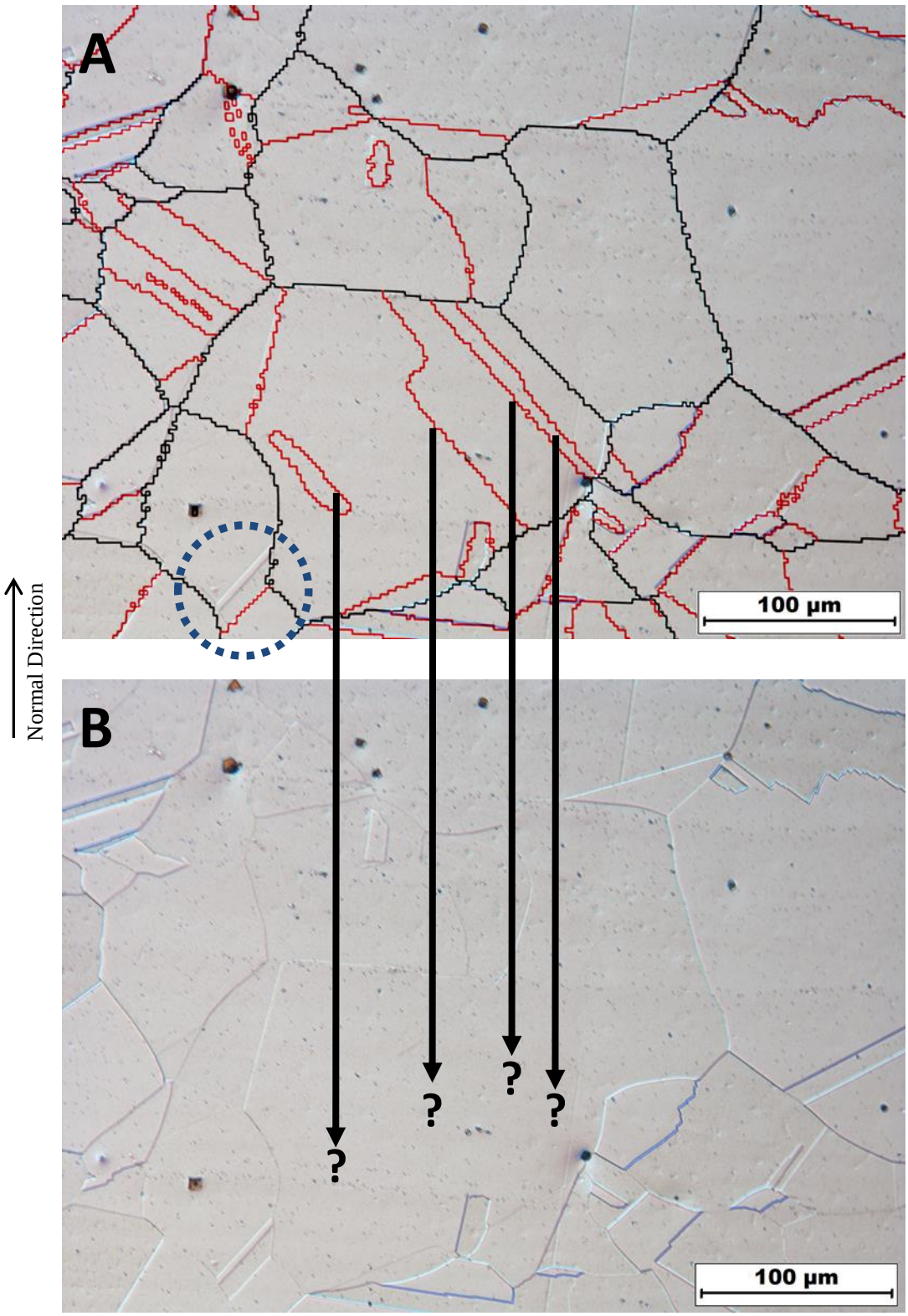


Figure 3.9 (A) EBSD map overlaying an optical image from the same region, (B), with lines indicating the absence of boundaries. The blue circle indicates the situation whereby the EBSD mapping resolution has failed to identify a pair of parallel Σ3 boundaries.

3.4 EBSD Mapping

The following section provides information related to the use of EBSD from an operator's point of view, and is somewhat restricted to the focus material, Alloy 800H, and the available SEM/EBSD hardware. A JEOL JSM 6100 scanning electron microscope coupled with an HKL Nordlys II electron backscatter detector was employed in this study, Figure 3.10.

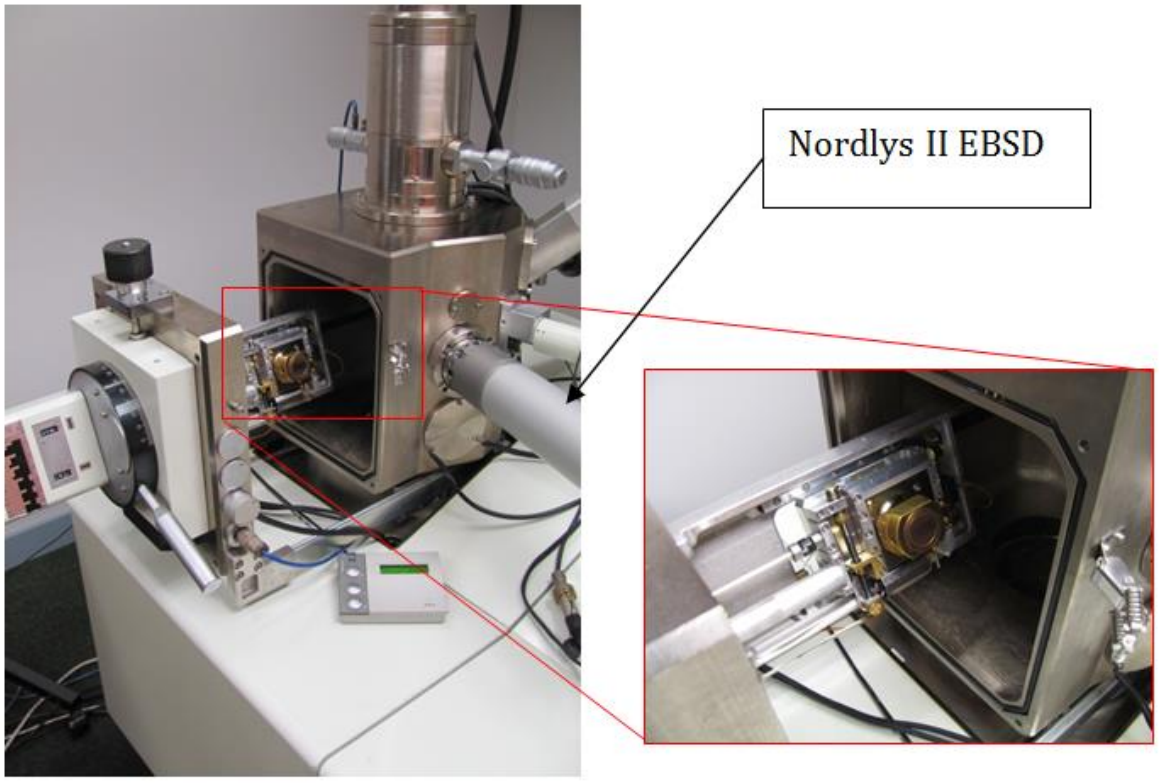


Figure 3.10 JEOL JSM 6100 scanning electron microscope coupled with an HKL Nordlys II electron backscatter detector.

3.4.1 Mapping Step Size

Step size is the most important parameter of the EBSD process because it sets the map resolution and determines the minimum sized feature that can be identified. The minimum sized features of interest were sets of parallel Σ3 boundaries, shown in Figure 3.11. When step size approaches the distance (width) between a pair of parallel Σ3 boundaries, either fragmentation occurs or the boundary is missed entirely. The fragmented appearance of Σ3 boundaries on an EBSD map may affect the quantification of boundary types.

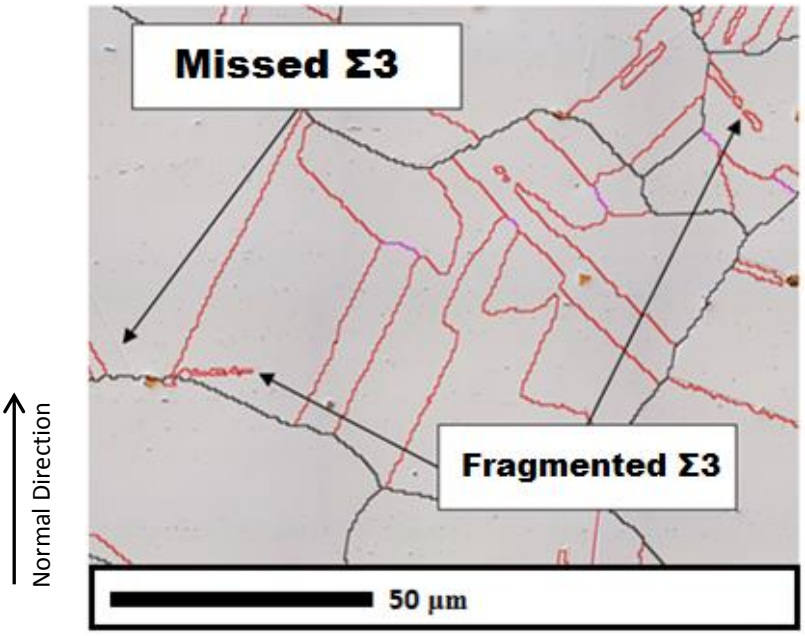


Figure 3.11 EBSD map overlaying an optical micrograph illustrating thin Σ3 regions acting as limiting factors for mapping resolution.

The distribution of different boundary types in the grain boundary network may be communicated as an absolute value, a length fraction, or a number fraction. In the current study, boundaries were identified as coherent Σ3 (twin), non-coherent Σ3s (Σ3 boundaries not identified as coherent twins), Σ9, or Σ27 boundaries. All other boundaries were considered random high angle grain boundaries (RHGB).

The length fraction is defined as the ratio between the total length of each boundary type, l_{BT} , and the total length of all HGBs, l_{HGB}. The length fraction, Lf_{BT}, for each boundary type is given by:

$$Lf_{BT} = \frac{l_{BT}}{l_{HGB}}$$

Equation 3.2

The number fraction is the ratio between the total number (absolute value) of boundaries of each type, n_{BT}, and the total number of HGBs, n_{HGB}. A boundary is identified from an EBSD map as a length of boundary connecting two triple points. The number fraction, Nf_{BT}, for each boundary type is given by:

$$Nf_{BT} = \frac{n_{BT}}{n_{HGB}}$$

Equation 3.3

Alexandreanu and Was [127] conducted a study into the number of boundaries requiring analysis in order to achieve a desired fractional error. Equation 3.4 describes the number of boundaries required, N, to achieve a fractional error, f_{ER}, for a measured number fraction, P.

$$N = \frac{1}{(f_{ER})^2}\frac{1-P}{P}$$

Equation 3.4

For example, if the measured number fraction of Σ3 boundaries was P = 0.30, and an error of no greater than $f_{ER} = 0.05$ was desired, a total of approximately 1,000 HGBs would need characterizing. For this study a minimum of 1,000 high-angle grain boundaries were characterized per sample.

Figures 3.12 to 3.14 show the gradual appearance of Σ3 boundaries at progressively finer step sizes for an Alloy 800H sample. By evaluating the change in the number of boundaries at each step size, an understanding is developed of how anomalies caused by mapping resolution affect the final grain boundary statistics. The current analysis is limited to Σ3 boundaries and, for the moment, all other boundaries shall be considered RHGBs.

As the step size is reduced from 6µm to 4µm, Figure 3.12 shows the appearance of a one sided Σ3 boundary. At a step size of 6µm, Figure 3.12(A), only the RHGB network is present. When the step size is reduced to 5µm, Figure 3.12(B), a Σ3 boundary appears. While the step size is sufficiently small to identify the Σ3, a mapping anomaly exists whereby an additional triple point is created due to the proximity of the Σ3 boundary to the RHGB. The consequence of this anomaly is that an additional Σ3 boundary will be quantified. In comparison, Figure 3.12(C) shows that at a step size of 4µm the boundary can be accurately observed with the final result being *one* additional Σ3 and two additional RHGB.

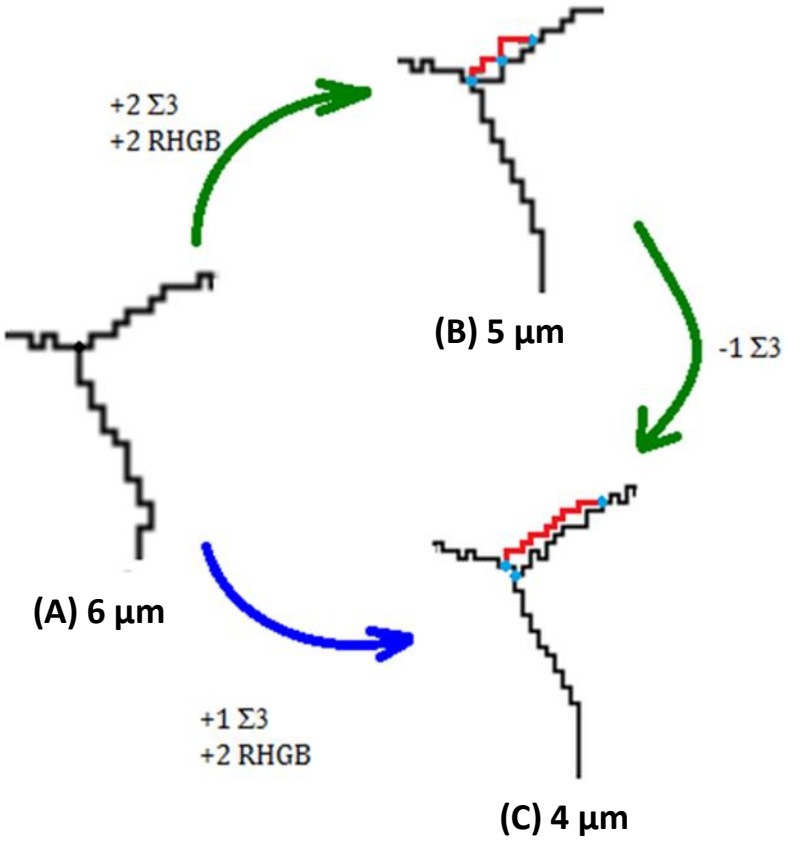


Figure 3.12 Appearance of a one sided Σ3 boundary.

(A) Three RHGB forming a triple point. Step size = 6µm.

(B) One sided Σ3 boundary appearing across RHGB triple point. A mapping anomaly has resulted in the formation of a false triple point. Step size = 5µm

(C) One sided Σ3 boundary present with no mapping anomaly. Step size = 4µm.

Figure 3.13 shows the appearance of a set of complete parallel sided Σ3 boundaries as the step size is reduced from 6µm to 3µm. The successful identification of the boundaries when a step size of 3µm is used, (D), resulted in an additional two Σ3 boundaries and three RHGBs. Figures 3.13(C) and 3.13(D) show elements of the Σ3 boundaries appearing, but with a resolution which is insufficient to reveal them completely. The Σ3 boundary's fragmented appearance results in over-quantifying Σ3 boundaries and under-quantifying RHGBs.

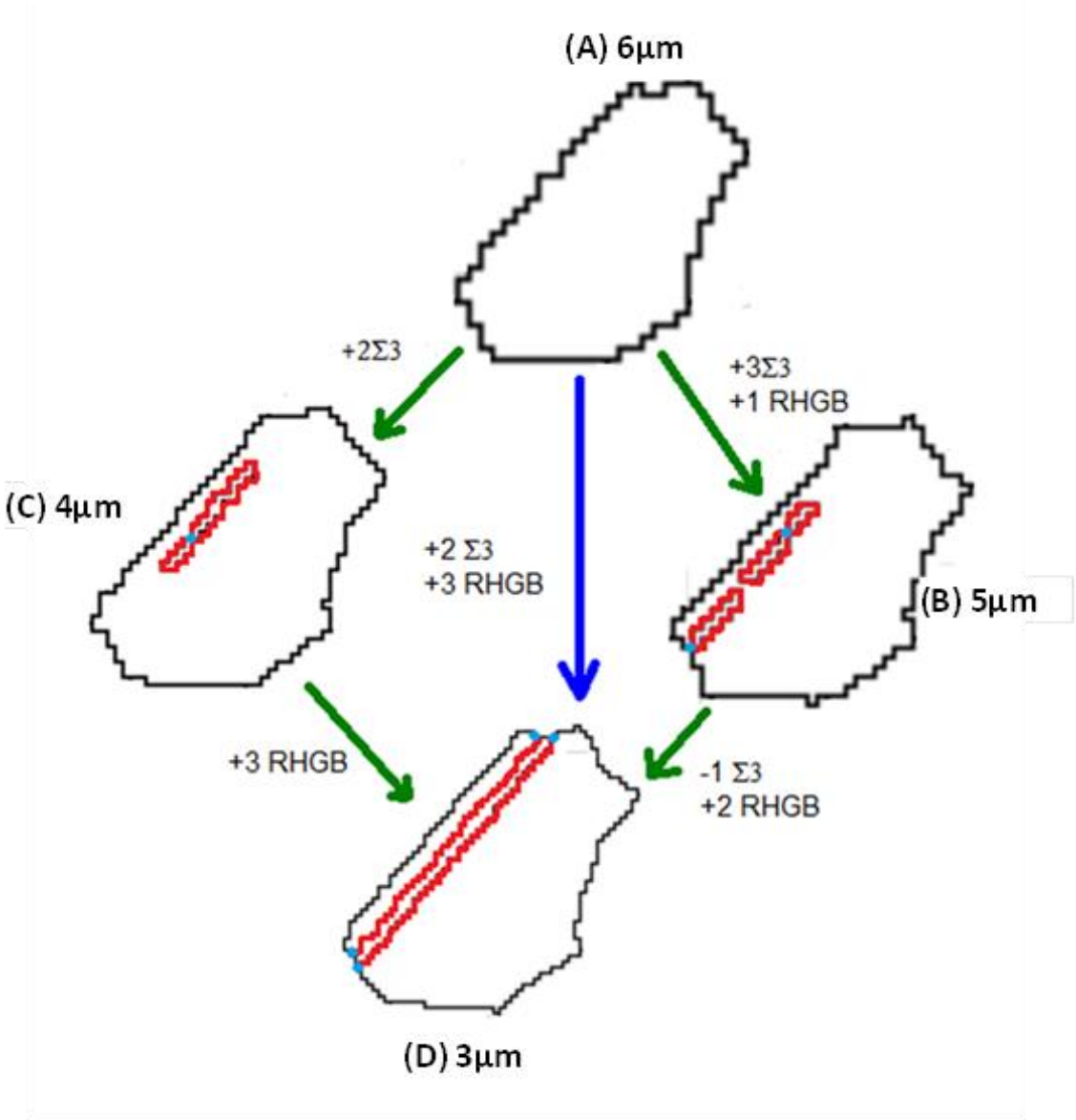


Figure 3.13 Appearance of a set of complete parallel sided Σ3 boundaries.

(A) Grain defined by RHGBs. Step size = 6µm

(B) Parallel sided Σ3 boundaries appearing. Portion of the boundary attached to the RHGB, but only at a single point. Two island Σ3s present. Step size = 5µm.

(C) Parallel sided Σ3 boundaries still not completely revealed. Step size = 4µm.

(D) Complete parallel sided twin identified in EBSD map. Step size = 3µm.

Figure 3.14 shows the appearance of an incomplete parallel sided Σ3 boundary as the step size is reduced from 7µm to 3µm. If the Σ3 boundary appears completely, Figure 3.14(E), the addition of one Σ3 and two RHGB occurs. Figure 3.14(B) and 3.13(D) are further examples of incorrectly revealed Σ3

boundaries resulting in an erroneous quantification of boundary types. Figure 3.14(C) is an example of where no indication of a mapping anomaly is observed; however, at a higher resolution of 4µm, an additional boundary element is revealed. This is later shown to be part of the original boundary, Figure 3.14(E).

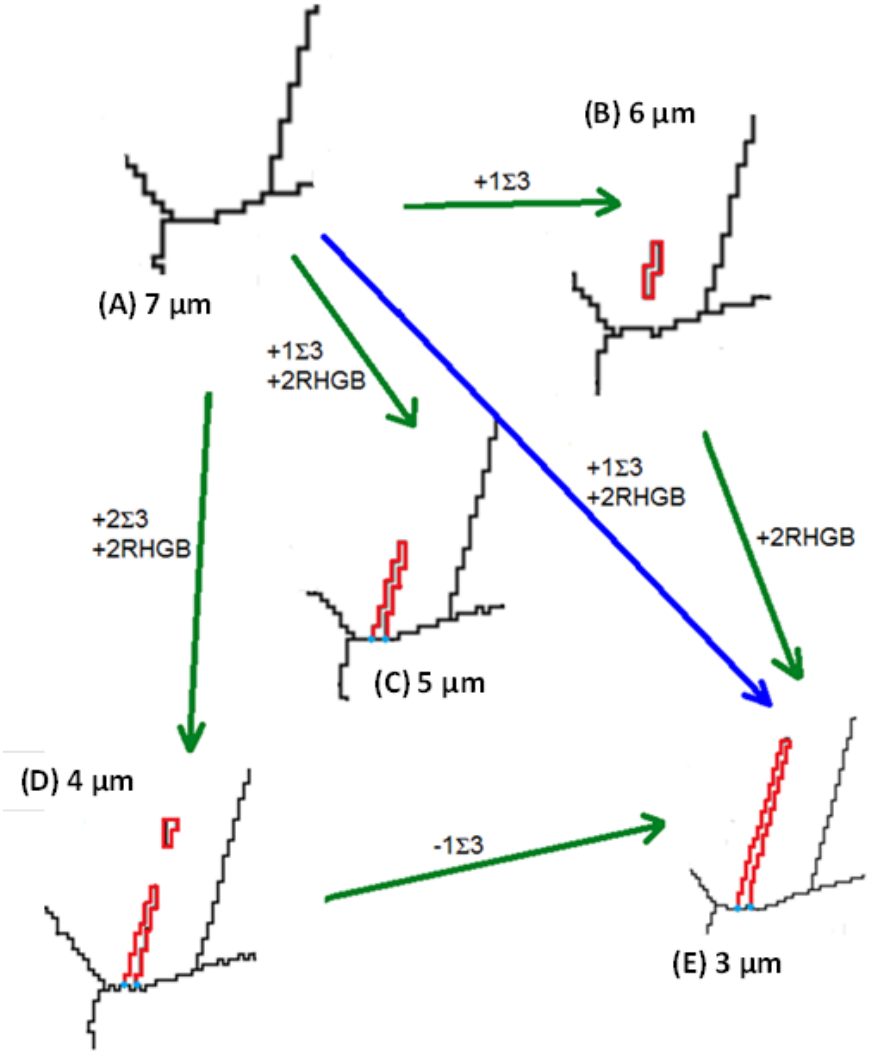


Figure 3.14 Appearance of an incomplete parallel sided Σ3 boundary.

(A) Section of the RHGB network. Step size = 7µm.

(B) Parallel sided Σ3 boundary appearing. Region currently exists disconnected from the RHGB network (island). Step size = 6µm.

(C) Incomplete parallel sided Σ3 boundary identified on the EBSD map. Step size = 5µm.

(D) Island Σ3 boundary appearing at the end of the incomplete parallel sized Σ3 boundary. Step size = 4µm.

(E) Island Σ3 appearing in (D) connects to the existing incomplete parallel sided Σ3 boundary. Step size = 3µm.

The influence of step size on the quantification of Σ3 boundaries is assessed by EBSD mapping an area on a sample at varying step sizes. Pande [96] showed for Nickel that the width between two twins increased with an increase in average grain size. Pande, from optical micrographs, measured twin width as the distance between two consecutive intersections of a random line by boundaries identified as twins. Therefore, because the distance between twin boundaries (a subset of Σ3) is a function of average grain size, it seems reasonable that a suitable step size for EBSD mapping should also be a function of average grain size. Two Alloy 800H samples representing different grain sizes, Tables 3.2 and 16.3, were selected for EBSD mapping. Six EBSD maps of the same area were produced using step sizes between 2 and 8μm. The purpose of the study was to identify how average grain size and the number of Σ3 boundaries varied with step size.

Table 3.2 Average grain sizes and boundary type measurements for Sample 1 mapped at step sizes 2μm to 7μm.

Sample 1						
Step Sizes (μm)	***7***	***6***	***5***	***4***	***3***	***2***
Average Grain Size ***(μm^2)***	73.1	69.2	66.9	63.8	61.8	60.1
Number of Σ3s	504	542	639	715	812	903
Number of RHGB	1567	1698	1780	1891	2028	2128
Number Fraction Σ3	24.3	24.2	26.4	27.4	28.6	29.8
Total Σ3 Length ***(mm)***	35.96	37.87	42.39	45.53	50.82	54.27
Total RHGB Length ***(mm)***	52.69	54.15	53.29	53.12	52.05	51.59
Σ3 Length Fraction	40.6	41.2	44.3	46.2	49.4	51.3

Table 3.3 Average grain size and boundary type measurements for Sample 2 mapped at step sizes 3 to 8μm.

Sample 2						
Step Sizes (μm)	***8***	***7***	***6***	***5***	***4***	***3***
Average Grain Size ***(μm^2)***	134.5	130.9	127.6	127.6	127.4	125.3
Number of Σ3s	350	374	392	412	435	465
Number of RHGB	1055	1082	1112	1114	1131	1210
Number Fraction Σ3	24.9	25.7	26.1	27.0	27.8	27.8
Total Σ3 Length ***(mm)***	50.77	52.67	54.42	55.93	58.85	61.79
Total RHGB Length ***(mm)***	62.34	62.41	62.53	60.70	59.71	60.96
Σ3 Length Fraction	44.9	45.8	46.5	48.0	49.6	50.3

Figures 3.15 and 3.16 show the number of Σ3 boundaries and RHGB identified from the EBSD maps produced for Samples 1 and 2 respectively. For Sample 1, the number of boundaries increased linearly with decreasing step size given by the equations:

$$n_{\Sigma 3} = -77\Delta + 1008$$

Equation 3.5

$$n_{RHGB} = -112\Delta + 2351$$

Equation 3.6

where n is the total number of boundaries and Δ is the step size. For Sample 2, the equivalent equations are:

$$n_{\Sigma 3} = -22\Delta + 512$$

Equation 3.7

$$n_{RHGB} = -26\Delta + 1263$$

Equation 3.8

In Sample 1, an additional 399 Σ3 boundaries and 561 RHGBs were revealed by decreasing step size from 7µm to 2µm, a ratio of 1.41 RHGBs for every 1 Σ3 boundary. Sample 2 showed a ratio of 1.35 RHGBs for every 1 Σ3 boundary. Figures 3.12 to 3.14 indicate that when a Σ3 boundary is accurately revealed (blue arrows), a theoretical increase of between 1.5 (one-sided, Figure 3.12, and incomplete parallel sided, Figure 3.14, Σ3) and 2 (complete parallel sided Σ3s, Figure 3.13) RHGBs is to be expected. The difference between the actual and theoretical Σ3/RHGB ratios is the due to boundary fragmentation, resulting in the identification of at least 15% more Σ3 boundaries than expected.

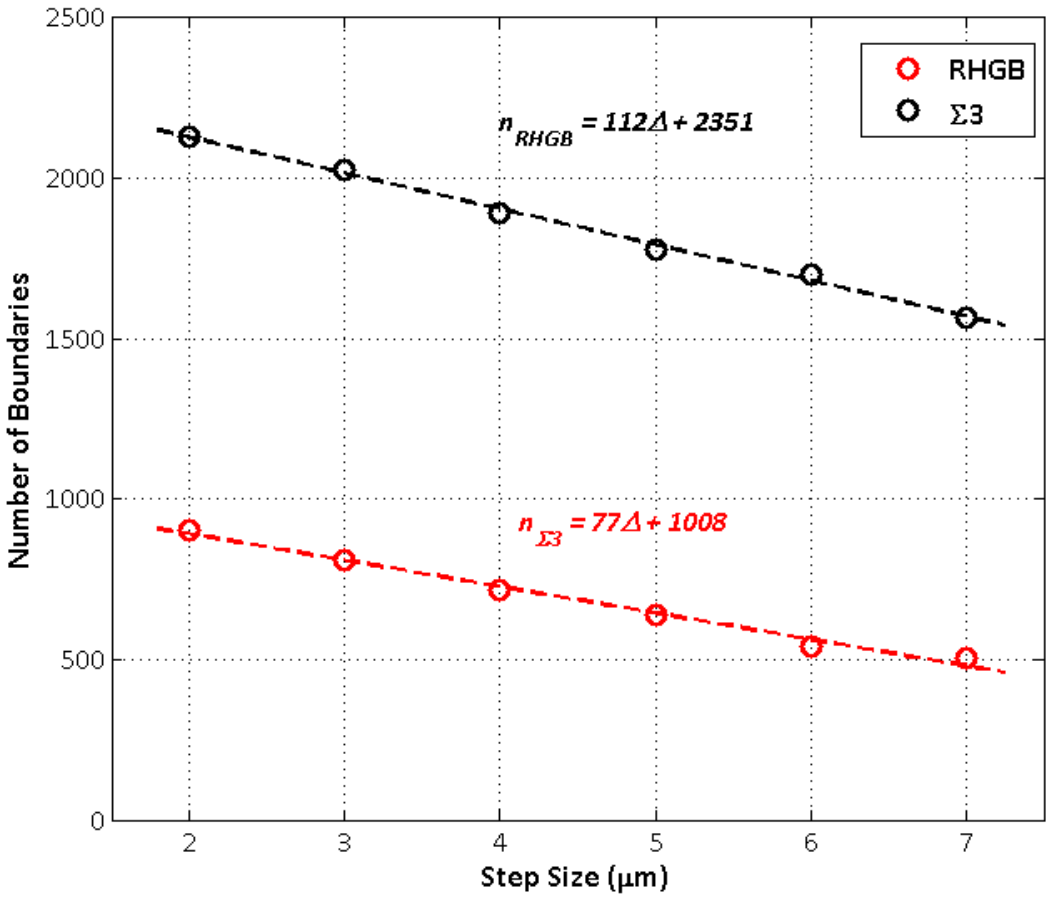


Figure 3.15 Number of Σ3s and RHGB at various mapping step sizes for Sample 1.

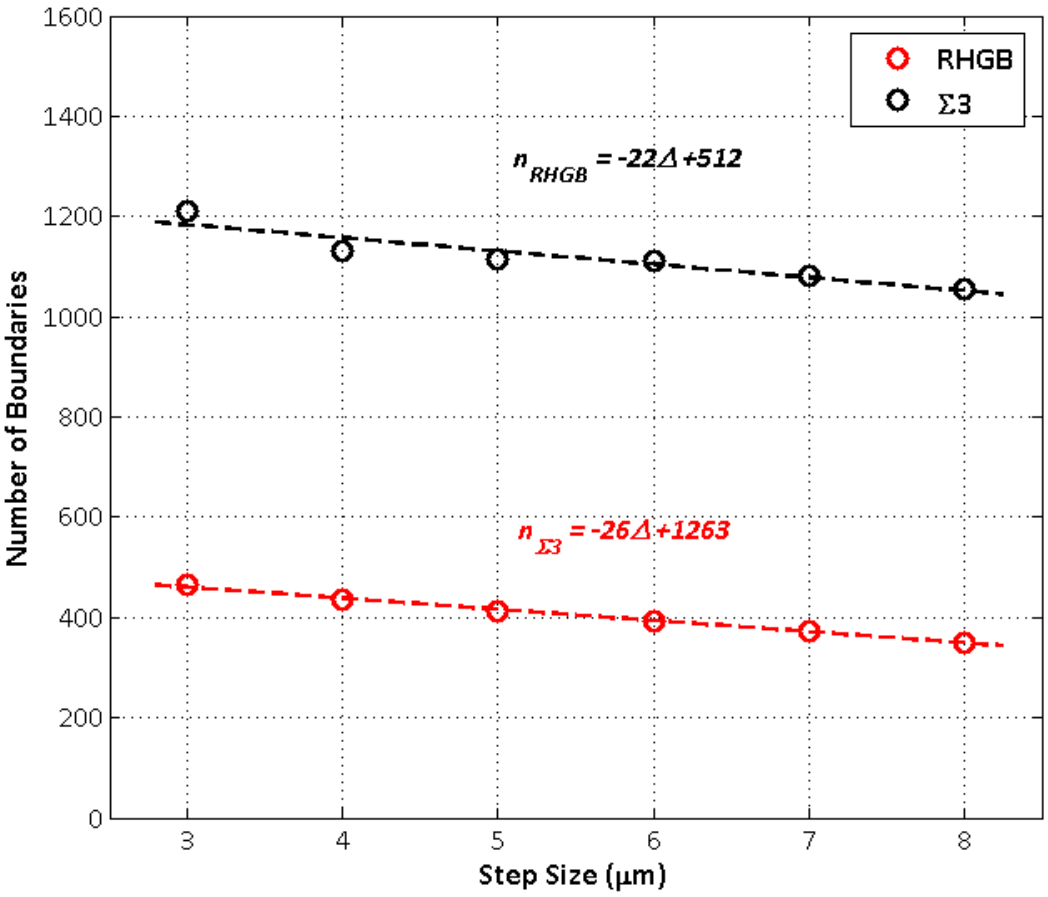


Figure 3.16 Number of Σ3s and RHGB at various mapping step sizes for Sample 2.

Figures 3.17 and 3.18 show the change in total boundary length for the boundary types as step size is decreased. For both samples, the total length of RHGB can be considered constant across all step sizes with less than 5% difference between the minimum and maximum RHGB boundary length. For Sample 1, the Σ3 boundary length increases linearly as step size is decreased given by the equation:

$$l_{\Sigma 3} = -3.8\Delta + 61.6$$

Equation 3.9

where $l_{\Sigma 3}$ is the Σ3 boundary length in mm and Δ is the step size. For Sample 2, the equivalent equation is:

$$l_{\Sigma 3} = -2.1\Delta + 67.5$$

Equation 3.10

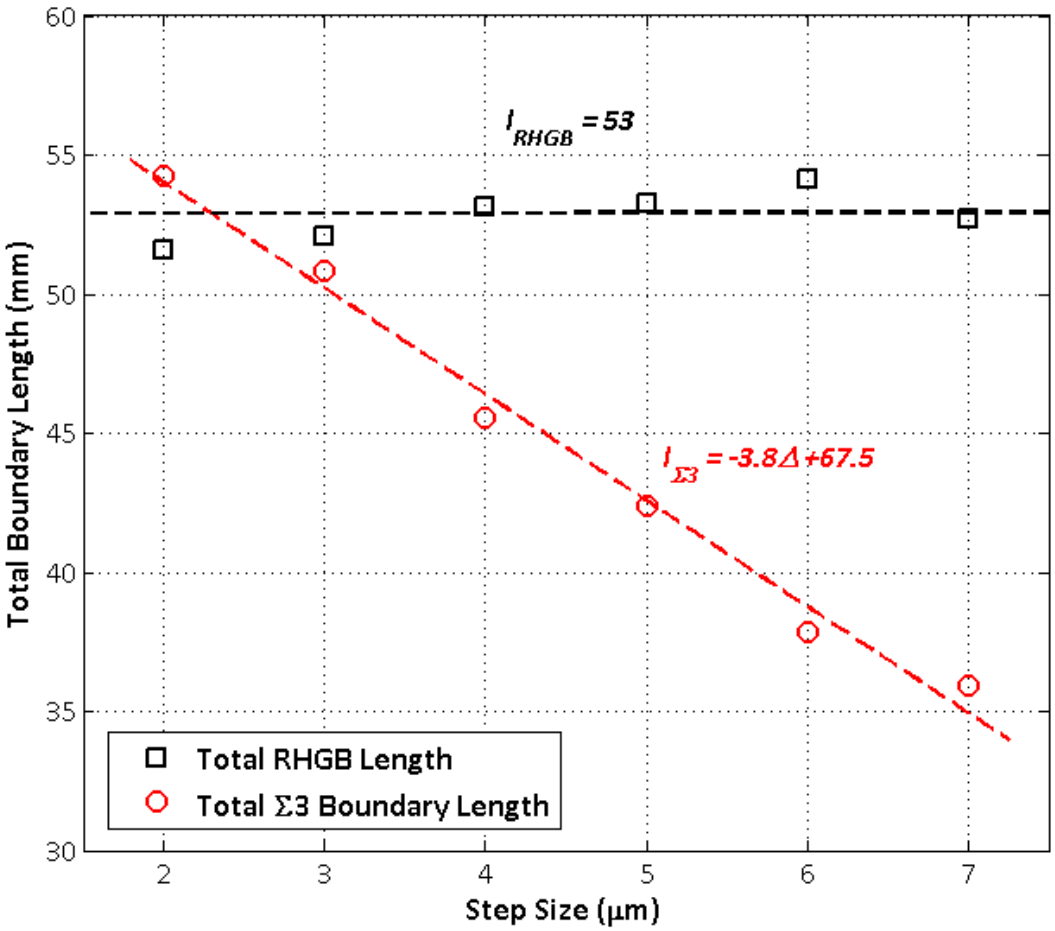


Figure 3.17 Total boundary length for Σ3 and RHGB at various mapping step size for Sample 1.

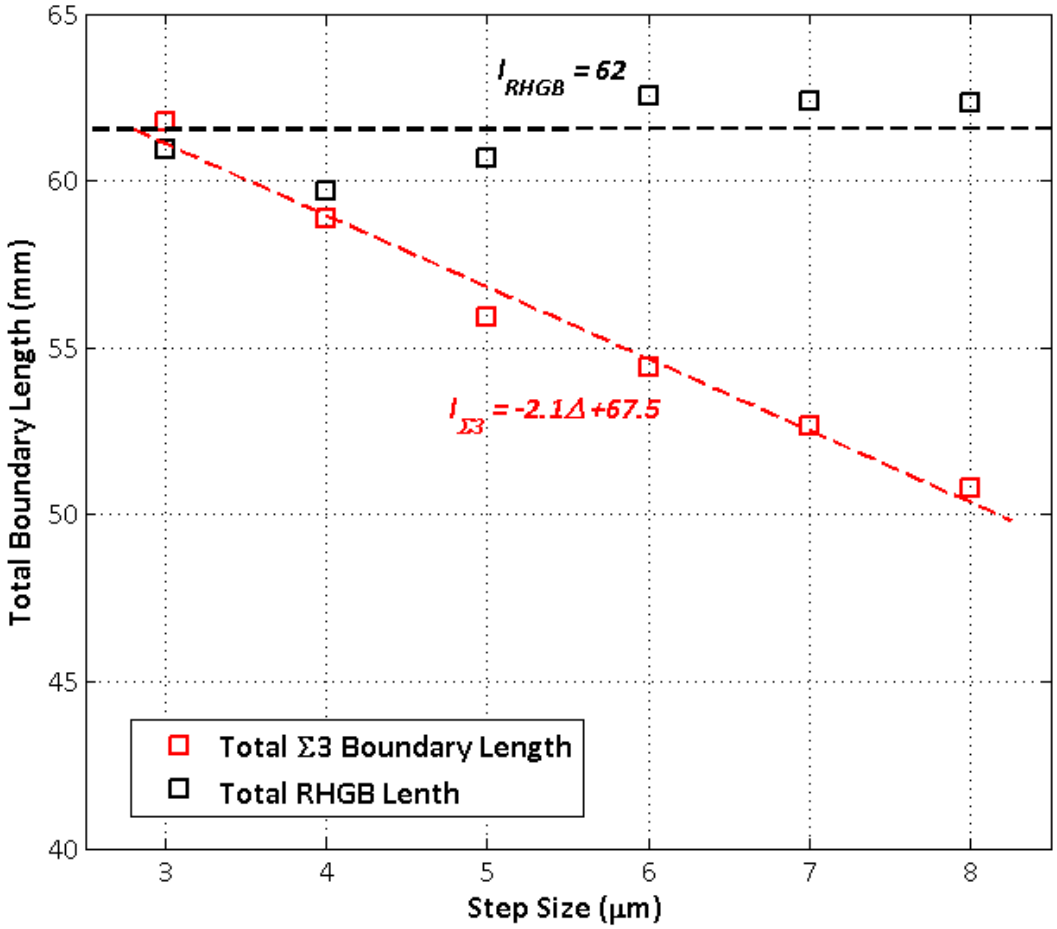


Figure 3.18 Total boundary length for Σ3 and RHGB at various mapping step size for Sample 2.

Figures 3.19 and 3.20 show the change in average grains size, Σ3 number fraction, and Σ3 length fraction for Samples 1 and 2 respectively. For both samples, the average grain size decreases as resolution increases because of smaller grains being revealed. For Sample 1, the Σ3 number and length fractions increase from 24.3% to 29.8% and 40.6% to 51.3%, respectively, as step size decreased from 7µm to 2µm. For Sample 2, the Σ3 number and length fractions increase from 24.9% to 27.8% and 44.9% to 50.3%, respectively, as step size decreased from 8µm to 3µm.

While all three types of measurement (absolute value, number fraction, and length fraction) can be used to quantify different boundary types within a microstructure, there are limitations to each method. Using the absolute number is the optimum type of measurement as it provides a direct measure to compare microstructures by communicating values such as Σ3s per grain or per unit area. The major limitation of using the absolute value is the sensitivity to the selected mapping step size. For example, Σ3s per grain increases from 1 to 1.4 for Sample 1 as step size is decreased from 7µm to 2µm.

The number and length fractions are relative measures and are less sensitive to the change in the absolute number of individual grain boundary types. Although the main advantage of the number fraction is that it communicates changes in the number of individual Σ3 boundaries, any change may be difficult to differentiate. For example, assume Sample 1 had 50% more Σ3 boundaries, i.e., for the 2µm step size map this would equate to an increase of 452 individual Σ3 boundaries. With the knowledge that the number of RHGB increases at a rate of 1.41 for every 1 Σ3, an increase in the number of RHGB to 2,765 is expected. Thus, a 50% increase in the number of Σ3 boundaries would equate to a number fraction of 32.9%. Taking into account that 15% of the total number of Σ3 boundaries may be the result of erroneous mapping, the number fraction may be as low as 29.5%, which is indistinguishable from the original sample with a Σ3 number fraction of 29.8%.

The length fraction is less affected by the presence of small fragmented boundary elements. A further advantage of the length fraction is a change in the number of Σ3 boundaries does not affect the total length of RHGBs, and as a consequence a more obvious change in fraction occurs. For example, again assuming Sample 1 has 50% more Σ3 boundaries, using an average Σ3 boundary length of 60µm (57.27mm/903 Σ3 boundaries, see Table 3.2), the total Σ3 boundary length increases to 81.43mm. Assuming no increases in RHGB length, the new length fraction will be 60.5% (an increase from 51.3%).

The obvious limitation of length fraction is the assumption that an increase in boundary type length is the result of an increase in the number of boundaries. However, based on the results from the current investigation this is a valid assumption because, for both Sample 1 and 2, the increase of both the number of Σ3 boundaries and the increase in total Σ3 boundary length produced linear relationships with respect to step size. In the current study the length fraction will be used to quantify the populations of different boundary types.

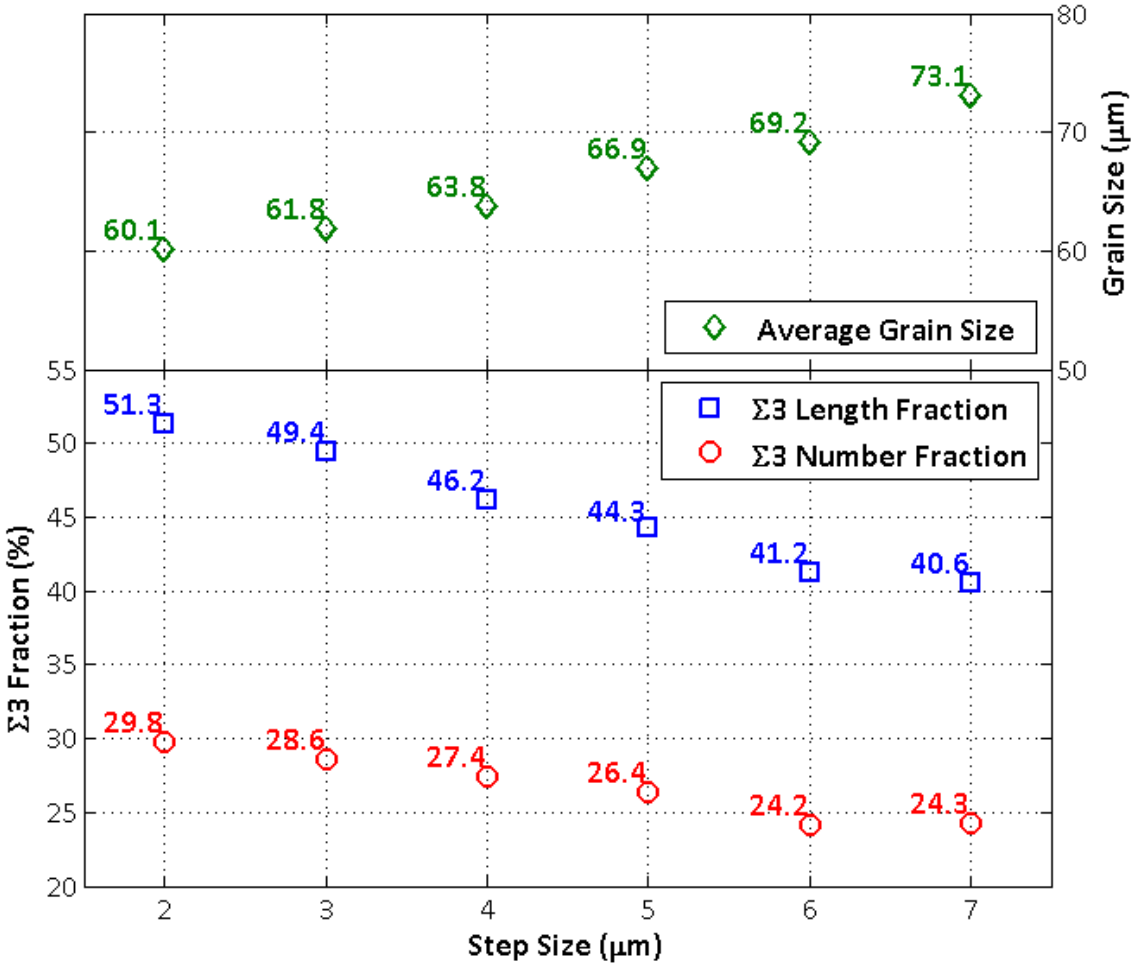


Figure 3.19 Average grain size and Σ3 number and length fraction at various step sizes for Sample 1.

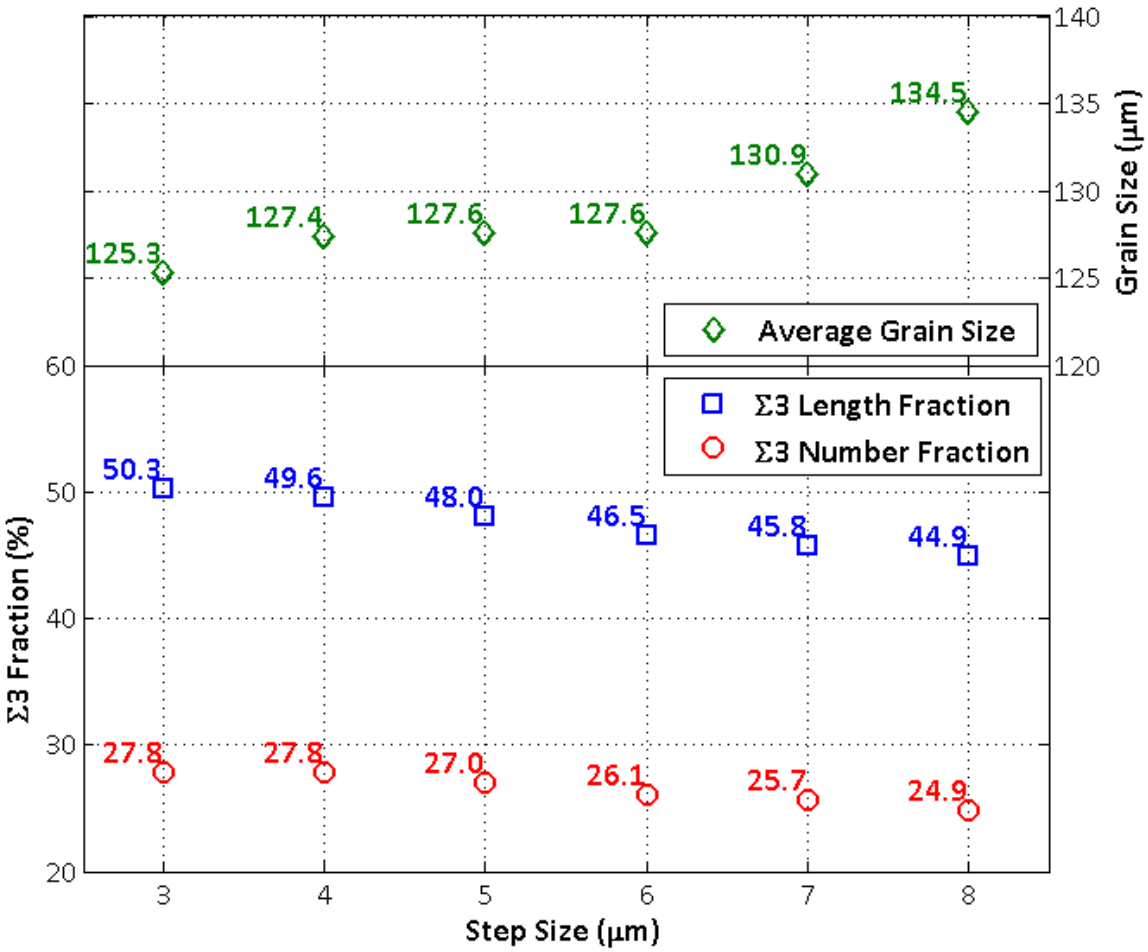


Figure 3.20 Average grain size and Σ3 number and length fraction at various step sizes for Sample 2.

The EBSD mapping step sizes for a range of average grain sizes is given in Table 3.4. These step sizes were determined by calculating a *theoretical* Σ3 length fraction. The theoretical total Σ3 length exists as Δ approaches 0. For Sample 1 this gives a theoretical Σ3 length fraction of $Lf_{\Sigma 3} = 53.8\%$, and for Sample 2, $Lf_{\Sigma 3} = 52.2\%$. A step size is selected based on the resulting $Lf_{\Sigma 3}$ having a difference of no greater than 5% from the theoretical value. For Sample 1, representing the average grain size range 50 to 100µm, the suitable step size is 2µm, and for Sample 2, representing the average grain size range 100 to 150µm, the suitable step size is 3µm. This analysis was repeated for three more samples representing additional average grain size ranges.

Table 3.4 Selected SEM and EBSD settings for mapping

Grain Size (μm)	Step Size (μm)	Minimum Map Size (mm^2)	Index Rate (pixels/sec)
10-50	1	0.25	10
50-100	2	1.00	20
100-150	3	2.25	20
150-200	4	4.00	20
200-250	5	6.25	20

3.4.2 Map Size and Index Rate

ASTM E2627-10 [121] recommends that at least 500 grains should be measured in total for the calculation of average grain size. For the current study, a minimum of 125 grains per EBSD map was targeted, with four maps produced per sample. Approximate map sizes for samples with various average grain sizes are presented in Table 3.4.

To index a diffraction pattern, time is required for the detector camera to capture the pattern imaged on the phosphor screen. The acquisition time is a function of the exposure time of the camera. If the camera exposure time is too short then the contrast in the pattern image is reduced.

For EBSD mapping, a short exposure time is ideal for fast index rates. Pattern contrast can be improved by increasing beam brightness, i.e. beam current. However, if beam current is too high, overlapping diffraction patterns are created that result in artifacts sited along the boundaries as shown in Figure 3.21. Table 3.4 shows the index rates (in pixels/second) for the different mapping step sizes. When a step size of 1μm is used, the beam current is reduced requiring longer exposure times (decrease in index rate) due to the decrease in brightness.

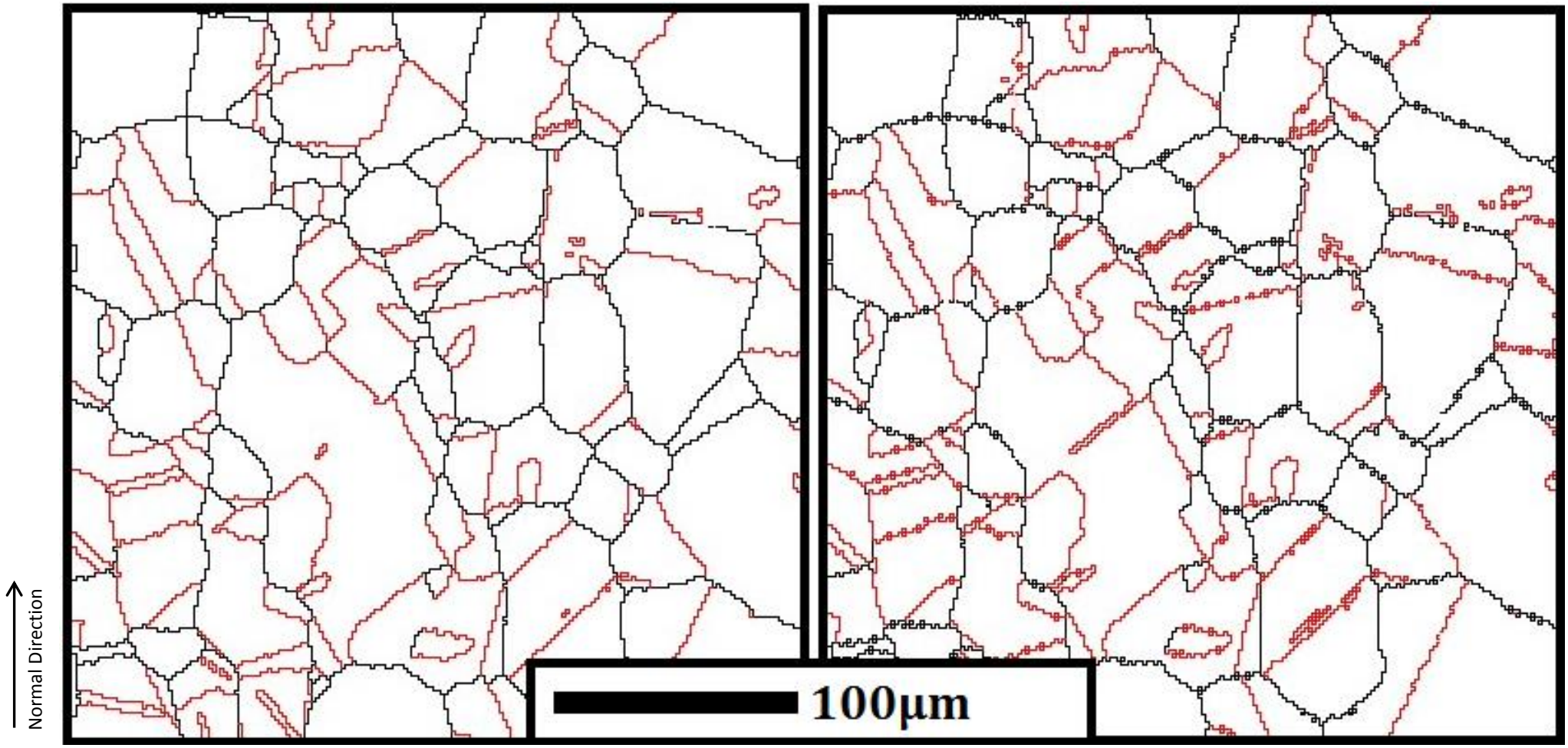


Figure 3.21 EBSD maps produced from the same area. The map on the right shows small pixel clusters along grain boundaries due to overlapping diffraction patterns.

3.4.3 Unindexed Pixels

Using the EBSD software, an algorithm is implemented assigning the average crystal orientation of the indexed pixels adjacent to a pixel that is unindexed. The software can implement the algorithm in two ways. If an iterative approach is taken, the unindexed pixels with the largest number of indexed nearest neighbours are solved first. With the single step approach, the unindexed points are assigned an orientation in the order that the map was produced (left-to-right/top-to-bottom) and assigning an orientation based on any number of indexed nearest neighbours. Because unindexed pixels are typically observed at grain boundaries, the boundary position may be influenced by which approach is used. The position of grain boundaries becomes important when reconstructing boundaries to identify potential coherent twins. For the current study, all unindexed pixels are solved using the iterative approach.

For the current study half of the EBSD maps produced had index rates in excess of 95% and no map was used where the index rate was below 90%.

3.5 Analysing EBSD Data

An algorithm (Appendix B) was developed for this study to analyse the orientation data provided by EBSD mapping, Figure 3.21. The algorithm was designed to provide measures for grain size and grain boundary character distributions (GBCD). Each input, process, and output will be explained in this section.

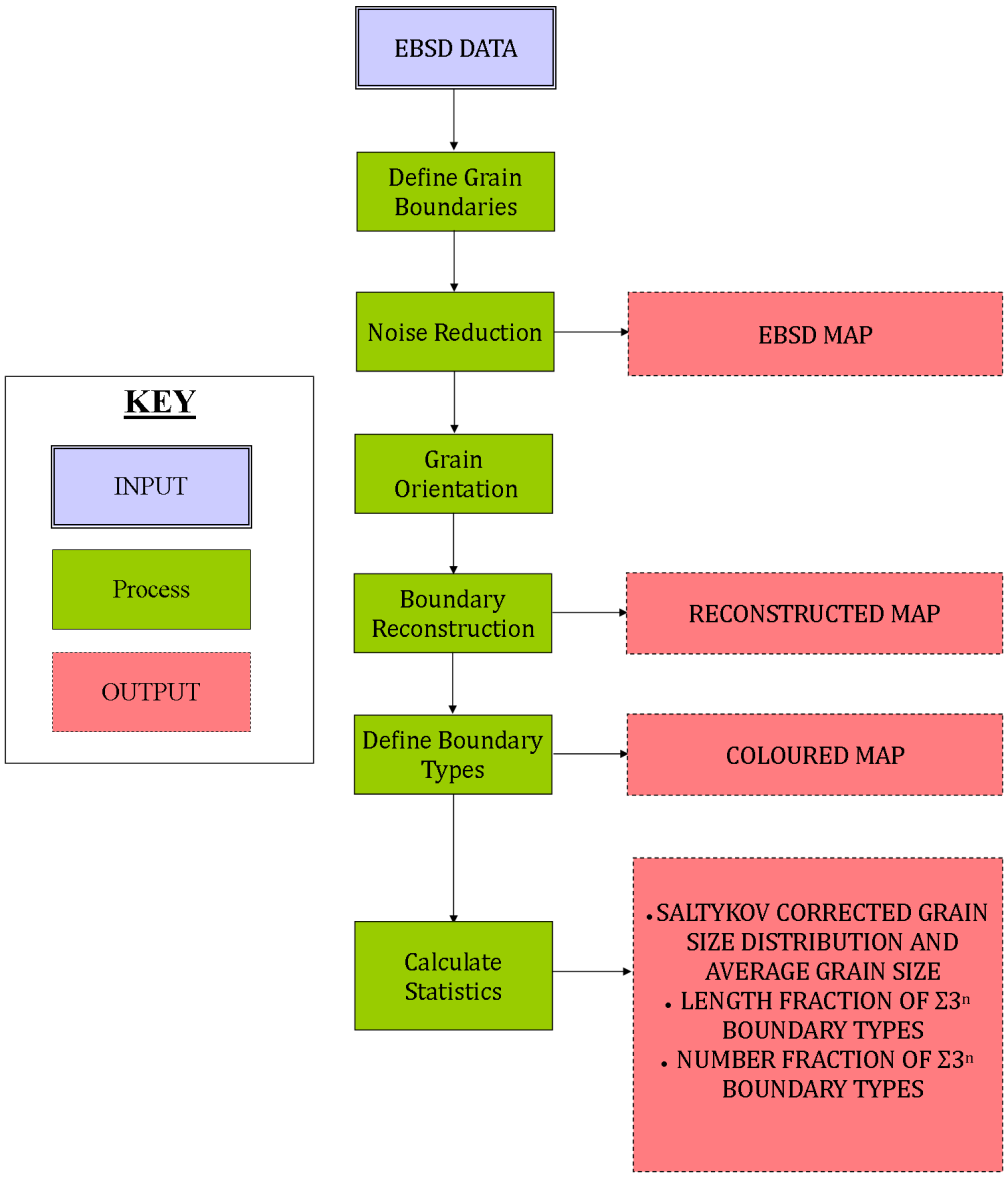


Figure 3.22 Outline of the inputs, outputs, and processes involved in the algorithm.

3.5.1 EBSD Data

The raw EBSD data is provided in the form of a text (.txt) file exported from the HKL EBSD software. The text data file contains information pertaining to the individual pixels (EBSPs), including pixel position on the mapping raster and the measured crystal orientation at that location (expressed with Bunge Euler Angles; $\varphi_1, \Phi, \varphi_2$). Crystal orientation at each EBSP is converted from Euler angles to quaternions using the method described in §2.3.2 (Equation 2.21).

3.5.2 Defining Grain Boundaries

Starting at the top left hand corner of the raster, the algorithm moves through all pixels (left to right) calculating the misorientation (Equation 2.21) with adjacent pixels. If the misorientation calculated is greater than 15˚, a HGB is said to exist between the pixels. Low-angle grain boundaries are ignored in this study as they have been shown to have a limited effect on creep performance under diffusional creep conditions [16]. Figure 3.23 is a section of the resulting matrix of identification numbers.

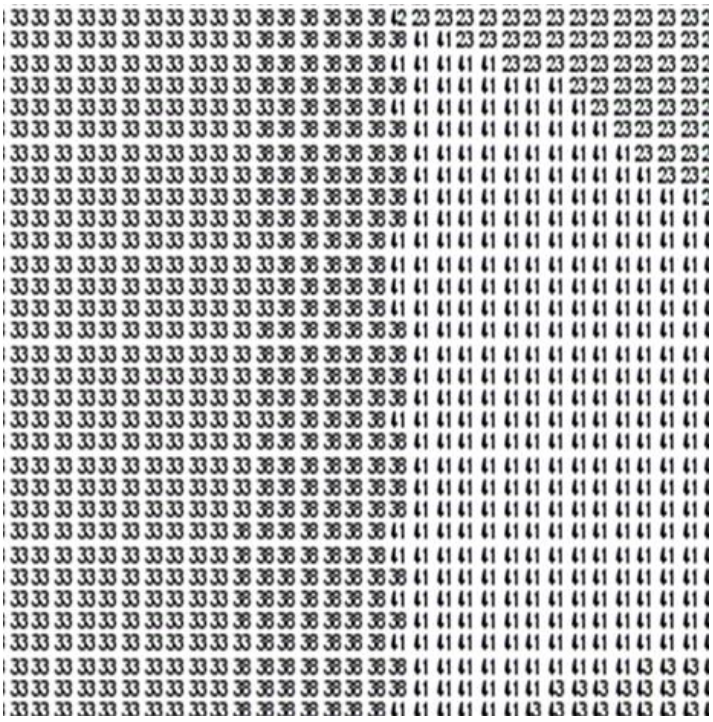


Figure 3.23 Section from an EBSD map with grains identified with a unique number.

3.5.3 Noise Reduction

The grain size distribution of the As-Received Alloy 800H pipe is shown in Figure 3.24. The distribution shown is not log-normal, as expected, with 45% of the grains populating the two smallest class sizes.

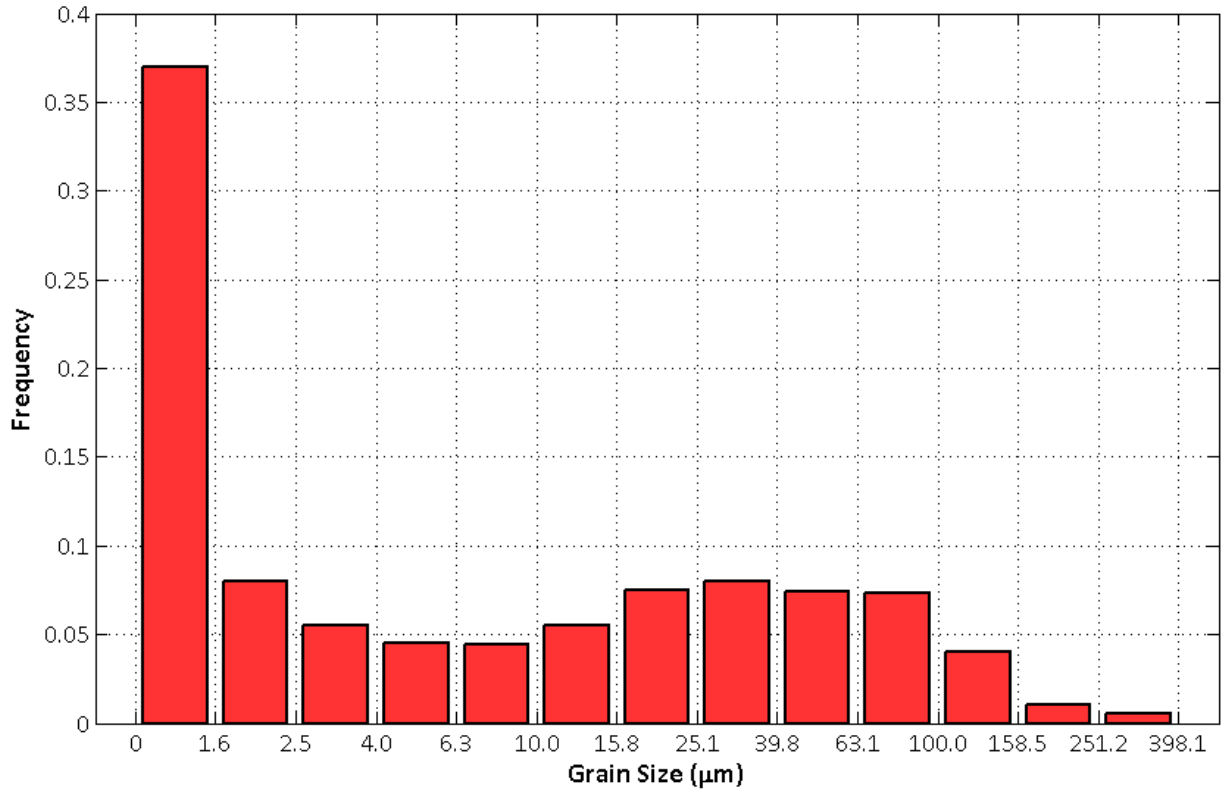


Figure 3.24 Grain size distribution for the As-Received pipe.

The distribution shown Figure 3.24 suggests that many of the smallest areas observed on the EBSD map may not represent a 2D section of a true grain. Their existence may be the result of secondary phases, fragmented Σ3 boundaries, or beam current producing overlapping diffraction patterns at grain boundaries, Figure 3.21. These pixel clusters representing false grains should therefore be removed from the maps by assimilating them into neighbouring grains. This process is called noise reduction.

A synthetic microstructure representing an equiaxed grain structure is shown in Figure 3.25(a). When sectioned, Figure 3.25(b), cross-sections of grains are shown adjacent to at least three neighbouring grains. Clusters of pixels located along grain boundaries, or within grain interiors, violate this assumption, with the exception being incomplete parallel twins and island twin morphologies.

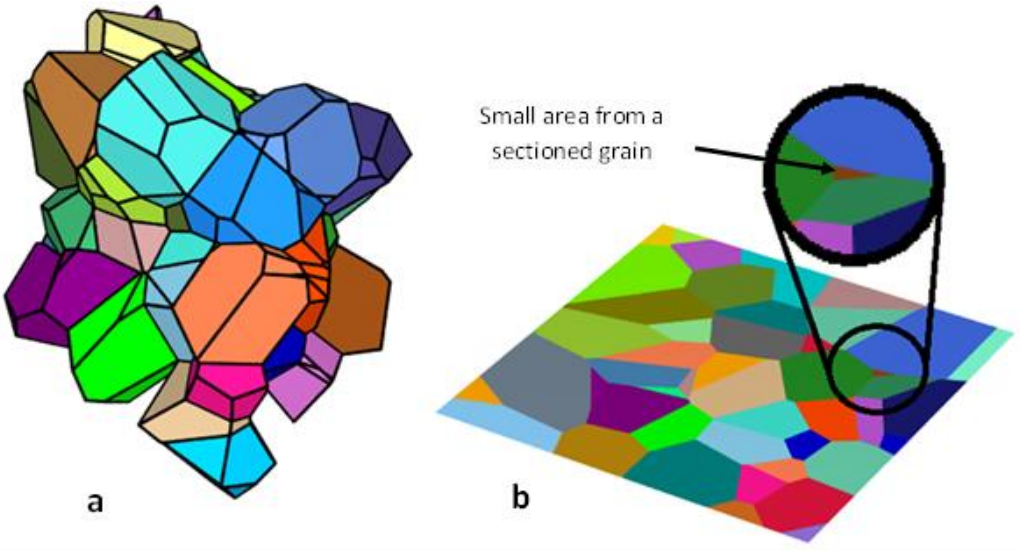


Figure 3.25 (a) Synthetic microstructure of equiaxed polyhedral with (b) a section showing a small grain located at the intersection of three grains.

Clusters ranging in size from 1 to 15 pixels were assessed to determine the proportion sited along the length of a boundary (side), within a grain interior (island), or positioned adjacent to three neighbouring grains (corner). The results are shown in Figure 3.26 for 31,000 pixel clusters analysed from 200 EBSD maps. The 200 EBSD maps were produced with step sizes ranging from 1µm to 6µm with average grain sizes between 20µm and 180µm.

The result shows that the smallest pixel clusters (1 to 5 pixels), that are principally responsible for producing the lower tail in Figure 3.24, are located primarily along the grain boundaries (side cluster) or within the grain interiors (island cluster). Similar analysis performed on the As-Received pipe sample showed that 30% of all the 'grains' observed on the EBSD maps were identified as side and island cluster 1 to 5 pixels in size. While it is a possibility that some of these clusters represent the cross-sections of actual grains, the majority exist due to boundary fragmentation, secondary phases, or the misindexing of diffraction patterns captured at grain boundaries.

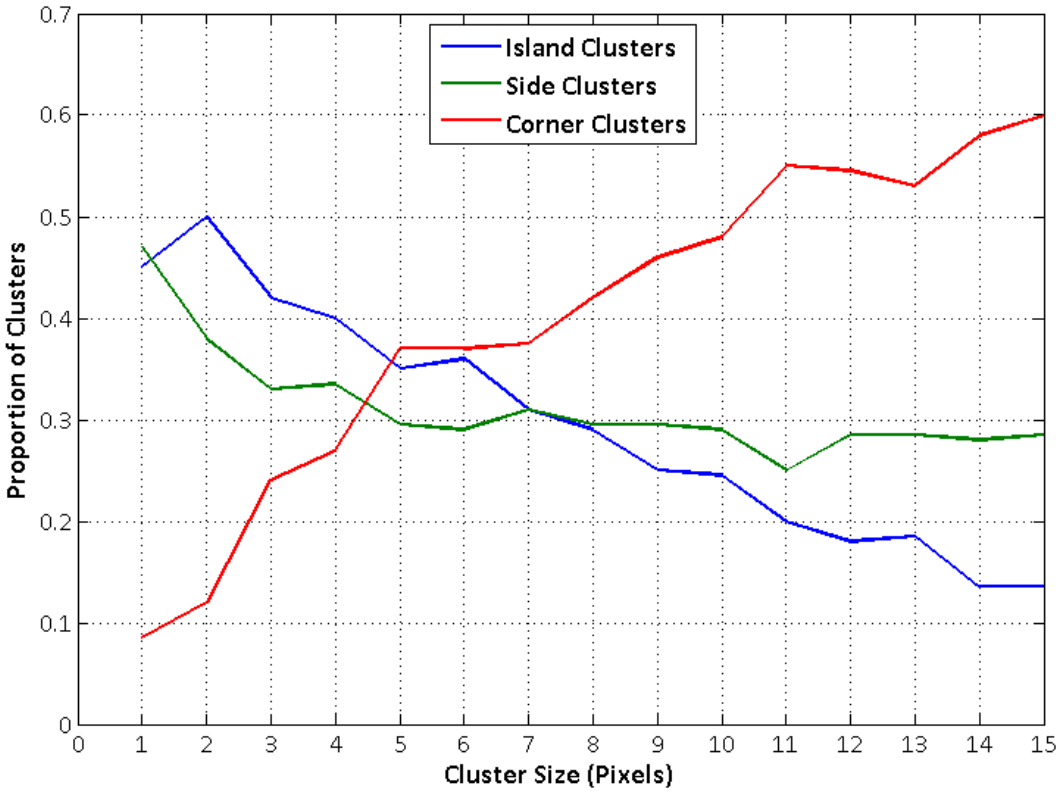


Figure 3.26 Proportion of clusters sited along the length of a boundary (side), within a grain interior (island), or with three neighbouring grains (corner) for 1 to 15 pixel clusters.

Figure 3.27 shows the reduction in the length of the lower tail of the grain size distribution after the small (1 to 5) pixel clusters were removed from the original EBSD map. Table 3.5 indicates that their removal resulted in a 79.4% increase in average grain size, and a 24.4% decrease in the standard deviation.

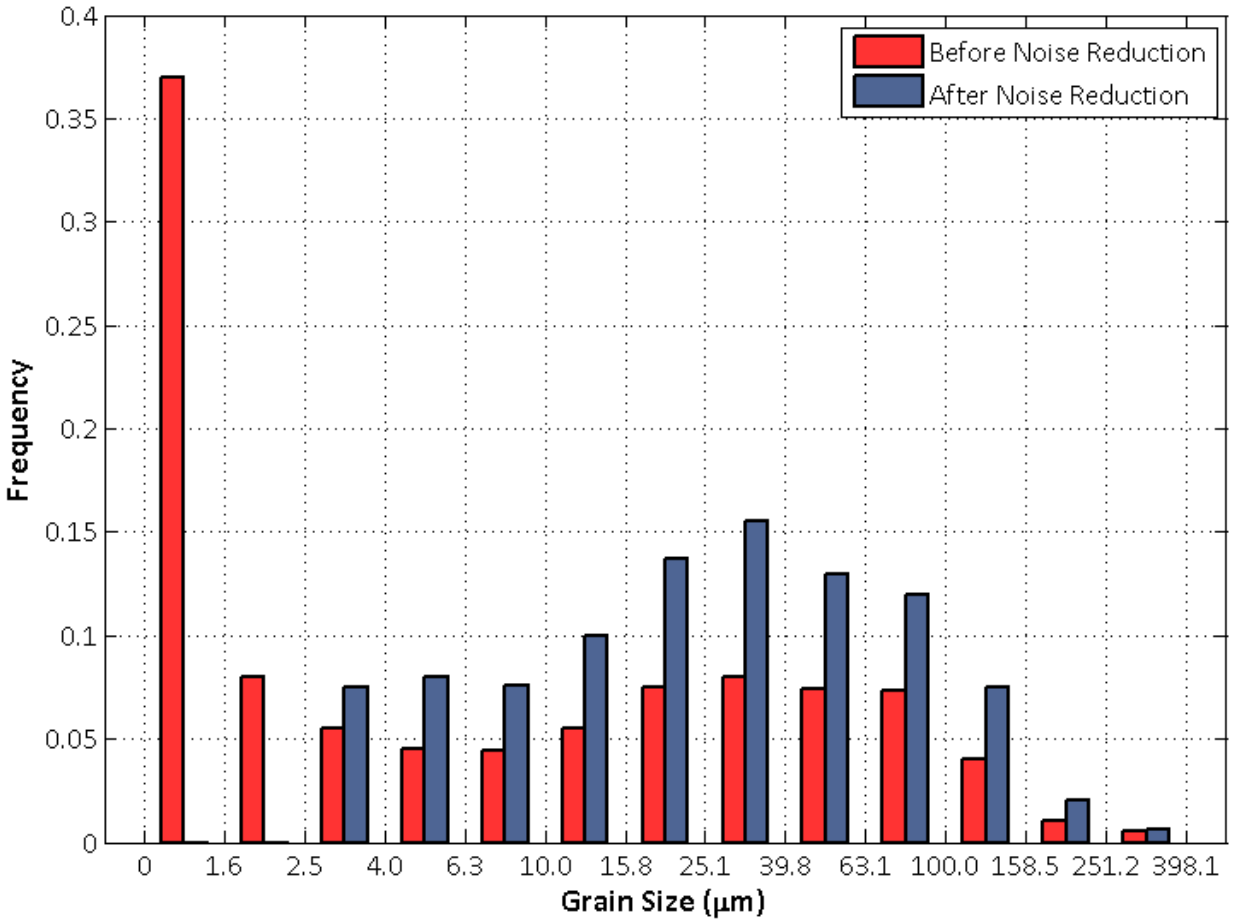


Figure 3.27 Grain size distribution for the as-received pipe comparing before and after 1 to 5 pixel clusters were removed.

Table 3.5 shows the change in boundary type statistics due to removal of the pixel clusters. The total boundary length was reduced by 1% due to the removal of the clusters resulting in a change in Σ3 length fraction of 1.3%, in contrast, the number fraction increased by 17.5%. The result further validates the earlier decision to use length fraction in the current study due to the sensitivity of the number fraction to the presence of small boundary elements.

Table 3.5 Grain size and boundary statistics before and after 1 to 5 pixel clusters were removed.

	Original EBSD Map	Noise Reduced EBSD Map	Difference Between Original and Noise Reduced EBSD Map
Average Grain Size (μm)	22.8	40.9	+79.4%
Standard Deviation[**]	1.152	0.871	-24.4%
Total Boundary Length (mm)	97.07	96.07	-1.0%
Σ3 Length Fraction (%)	56.0	55.3	-1.3%
Total Number of Boundaries	4666	3473	-25.6%
Σ3 Number Fraction (%)	24.6	28.9	+17.5%

The noise reduction process eliminates clusters up to five pixels by assimilating them into adjacent grains. Figure 3.28 illustrates the process for a three pixel cluster, identified by the number 2. In Figure 3.28(b), one of the pixels is highlighted in red and its eight nearest neighbour pixels in blue, orange, or green. The blue pixels belong to the grain contributing the highest number of nearest neighbours, the orange the least, and the green indicate pixels belonging to the cluster, and therefore not considered in the decision process. The cluster pixel is reassigned the grain identification number of the adjacent grain offering the highest number of nearest neighbour pixels; in this case, it was reassigned the grain identification number 1. The pixel is also assigned the average orientation of the nearest neighbour pixels. The process is repeated for all pixels within the cluster, Figures 3.28(c) and 3.28(d), with the final result shown in Figure 3.28(e). Figure 3.29 shows one pixel regions along a grain boundary that have been assimilated into adjacent grains.

[**] Equation 2.16

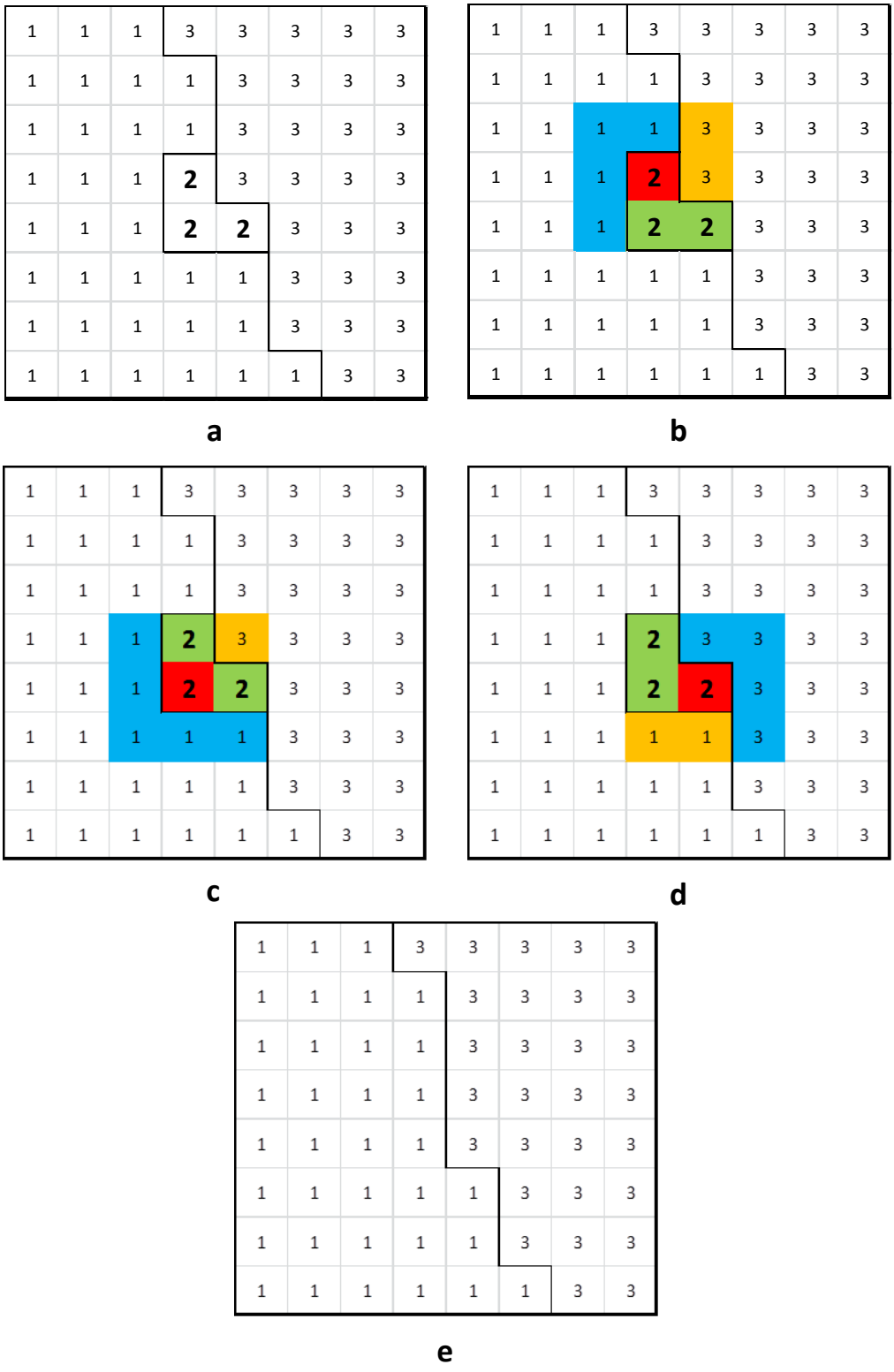


Figure 3.28 Nosie reduction removing small pixel clusters. (a) Identification of three pixel cluster for removal - number 2, (b-d) the adjacent grain contributing the largest number of neighbouring pixels is selected to replace the individual pixels within the cluster, (e) boundary after the removal of the cluster.

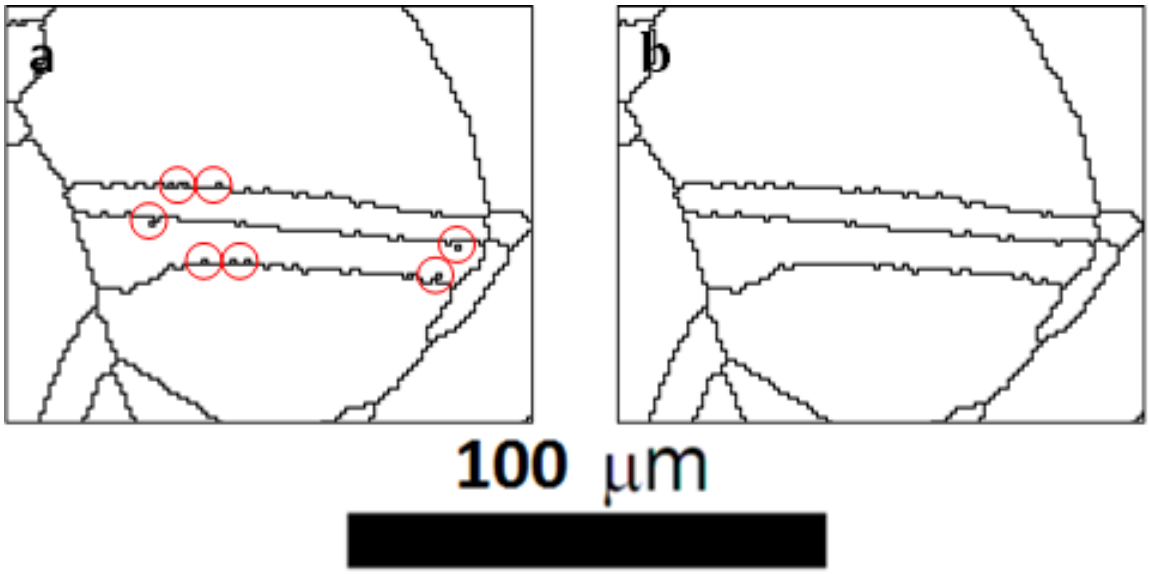


Figure 3.29 One pixel regions sited along the boundaries in (a) are removed from the EBSD map resulting in (b).

3.5.4 Average Grain Boundary Orientation

To determine the grain boundary type, the average orientation in the vicinity of each boundary is calculated. The algorithm employs a method similar to that of Wright and Larsen [128], that is, the pixels directly adjacent to the boundary are avoided due to possible lattice distortions. Occasionally, orientation variations were observed between grain centres and grain boundaries; therefore, only orientation measurements near the boundaries were selected.

Using the quaternion averaging methodology described by Cho and Rollet [129], the average orientation of the pixels in the vicinity of each grain boundary was calculated. The locations of the selected pixels were one pixel away from the boundary, Figure 3.30. In order to calculate the quaternion average, it is first necessary to minimise the angle between the relevant EBSP measurements. This process is performed by inspecting each of the 24 symmetry related equivalents for each EBSP and ensuring the orientation angles between all EBSPs are minimised. The average quaternion is then simply the arithmetic average of all relevant quaternions.

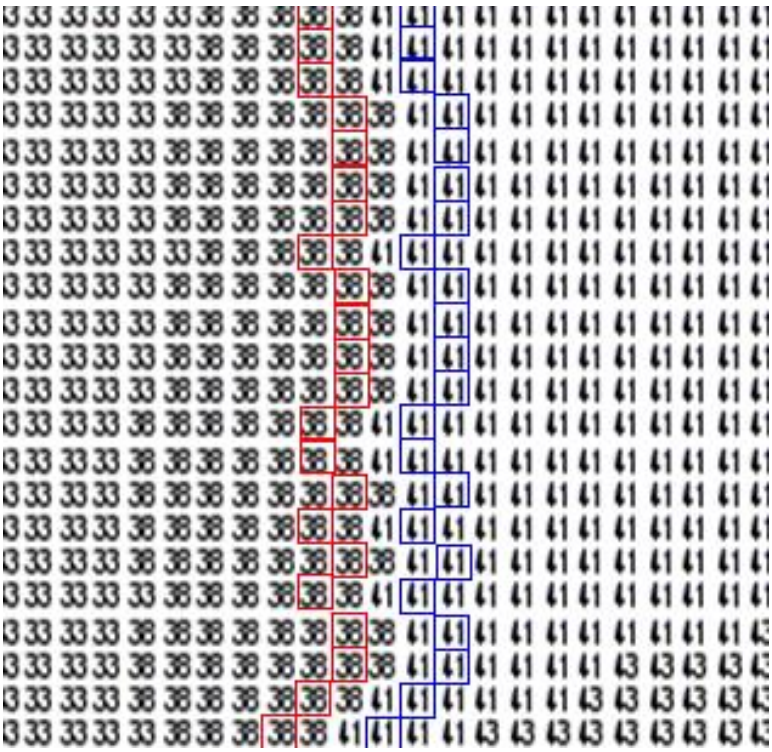


Figure 3.30 Illustration of the pixels that would be selected for averaging in Figure 3.23 for grains 38 (red) and 41 (blue) associated with boundary 38/41.

Analysis was performed on 500 grains to compare the average orientation of the pixels selected in the vicinity of a boundary and the average orientation for all pixels within the grain. An orientation difference of less than 1° occurred for 91% of the boundaries, and a maximum orientation difference of 8.6° was once recorded. The results indicate that, while the average orientation of the entire grain may accurately describe the orientation at a boundary, there are instances when the orientation difference is significant enough to influence subsequent analysis.

3.5.5 Boundary Reconstruction

Boundary reconstruction is a process which involves the replacement of the pixelated EBSD boundaries with straight line segments. The boundary reconstruction process, shown in Figure 3.31(a-c), is similar to that described by Wright and Larsen [128]. In Figure 3.31(a) the triple points of the red EBSD boundary trace are identified, and nodes placed at their locations. In Figure 3.31(b) an initial attempt is made at reconstructing the boundary by connecting the two nodes using a line segment. If the largest distance between the line segment and the EBSD boundary, δ, is greater than the reconstruction tolerance, then

a third node is placed on the EBSD boundary at the point where δ was measured. Now two line segments are used to accurately reflect the EBSD boundary morphology, Figure 3.31(c).

The reconstruction tolerance, δ, is given in multiples of mapping step size. Two-times the mapping step size has previously [122] been suggested as a suitable reconstruction tolerance and will therefore be employed in the current study. The result from the boundary reconstruction process performed on an EBSD map of Alloy 800H is presented Figure 3.32.

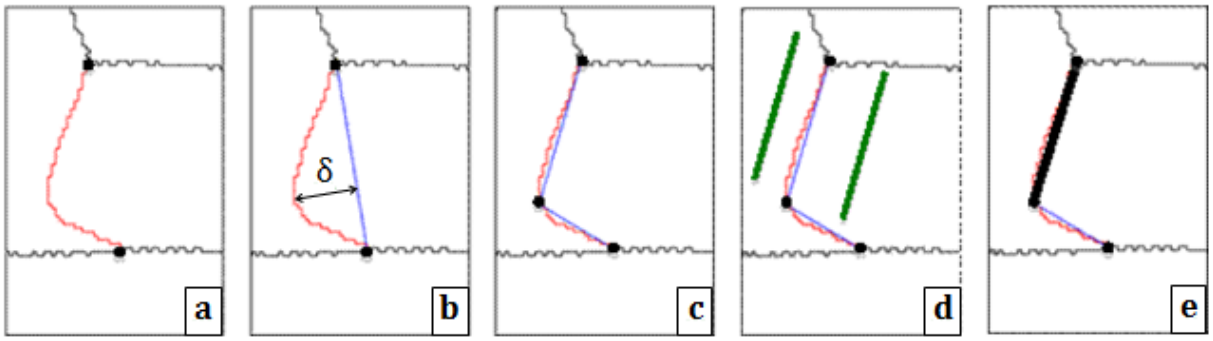


Figure 3.31 Example of boundary reconstruction and trace analysis.
(a) Red Σ3 boundary between two triple points/nodes.
(b) First reconstruction attempt (blue line).
(c) Reconstruction split into two line segments.
(d) {111} traces from adjacent grains in green.
(e) Coherent Σ3 boundary shown as a thick black line.

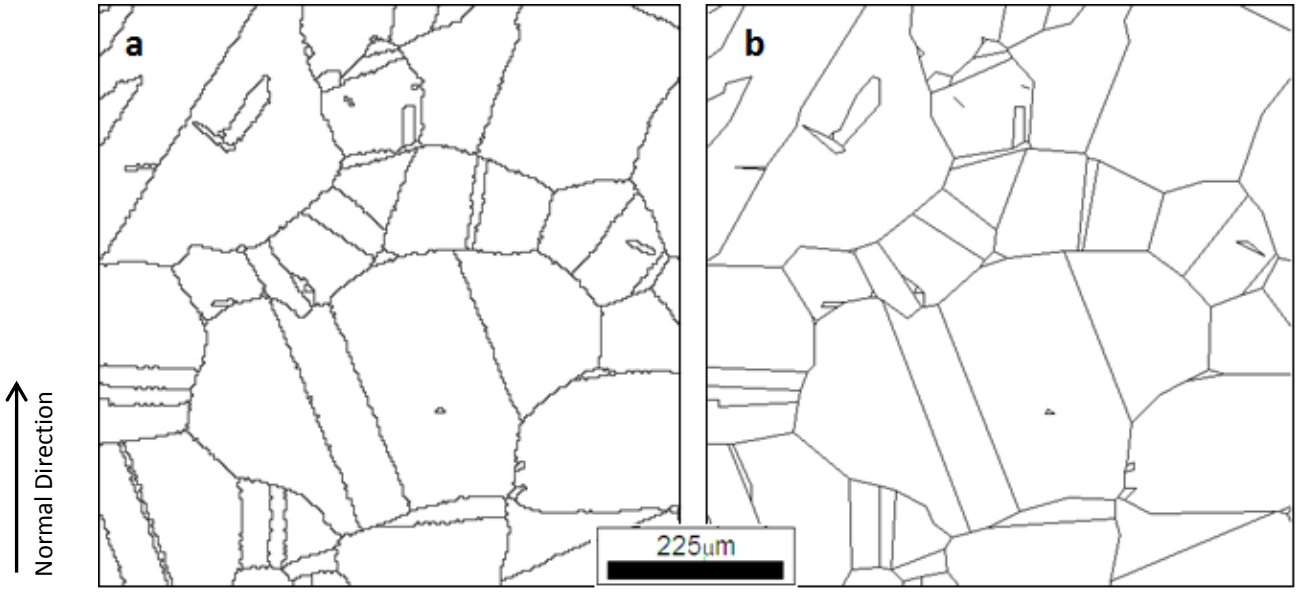


Figure 3.32 a) Grain boundary map produced using EBSD. b) EBSD boundaries reconstructed using straight line segments.

3.5.6 Define CSL Boundary Types

The misorientation angle, ω, and axis, $\hat{n}$, for each boundary can be calculated using the average grain orientations at adjacent boundaries. The rotation describing each grain boundary is compared with those describing the $\Sigma 3^n$ (n = 1, 2, and 3) boundary types, where ω and $\hat{n}$ is given in Table 3.6.

Table 3.6 Boundary misorientation angle and axis for $\Sigma 3^n$ (n = 1, 2, and 3) boundary types.

Σ Value	Misorientation Angle, ω	Misorientation Axis, $\hat{n}$	Brandon Criterion, v_m
3	60°	<111>	8.67°
9	38.94°	<110>	5.00°
27a	31.58°	<110>	2.89°
27b	35.42°	<210>	2.89°

Equation 3.11 gives the quaternion representing the misorientation of a $\Sigma 3^n$ boundary:

$$q_{\Sigma 3^n} = \left[\cos\frac{\omega}{2}, \sin\frac{\omega}{2}\hat{n}\right]$$

Equation 3.11

The orientation difference between a quaternion representing the misorientation of a boundary and a quaternion representing the misorientation of a $\Sigma 3^n$ is calculated. If the difference is less than the maximum allowable deviation, v_m, as determined by the Brandon criterion, Equation 2.24, the boundary is assigned the relevant Σ value.

3.5.7 Coherent Σ3 Line Segments

Trace analysis [130] is used to identify which $\Sigma 3$ line segments are coherent. A line segment is identified as coherent when it lies parallel to a {111} plane trace in each of the adjacent grains.

Figure 3.31(d-e) schematically demonstrates the trace analysis process. In Figure 3.31(d) the green lines represent the traces formed on the sectioned surface from {111} planes. If the {111} plane traces (green) are parallel to the $\Sigma 3$ line segment, the segment can be considered a coherent line segment, Figure 3.31(e).

Figure 3.33 is a pole figure representing the {111} interface normals of two grains (j = 1 & j = 2) adjacent to a Σ3 boundary (red). The unit vector, $\hat{l}$, tangent to the red Σ3 boundary is:

$$\hat{l} = [\cos\theta, \sin\theta]$$

Equation 3.12

where θ (0 ≤ θ < 180°) is the line segment angle measured from the horizontal of the EBSD map (x-axis) in an anticlockwise direction. Equation 3.13 calculates the normal vectors, $\hat{t}_{i,j}$, for the {111} interface planes with respect to the crystal orientations, q_j,, of the grains (j = 1 & j = 2). S_i represents the 24 symmetric operators given in §2.3.2.

$$\hat{t}_{i,j} = \left[\frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}}\ \frac{1}{\sqrt{3}}\right] \cdot q_j \cdot S_i$$

Equation 3.13

The vector $\hat{t}_{i,j}$ is projected onto the sectioning plane, proj$\hat{t}_{i,j}$, and the angle between the proj$\hat{t}_{i,j}$ and $\hat{l}$ is calculated. If the $\hat{t}_{i,j}$ projection is orthogonal to $\hat{l}$, then the {111} plane trace is parallel to the Σ3 line segment. The tolerance given to the orthogonal condition, α, is related to the measurement error on θ.

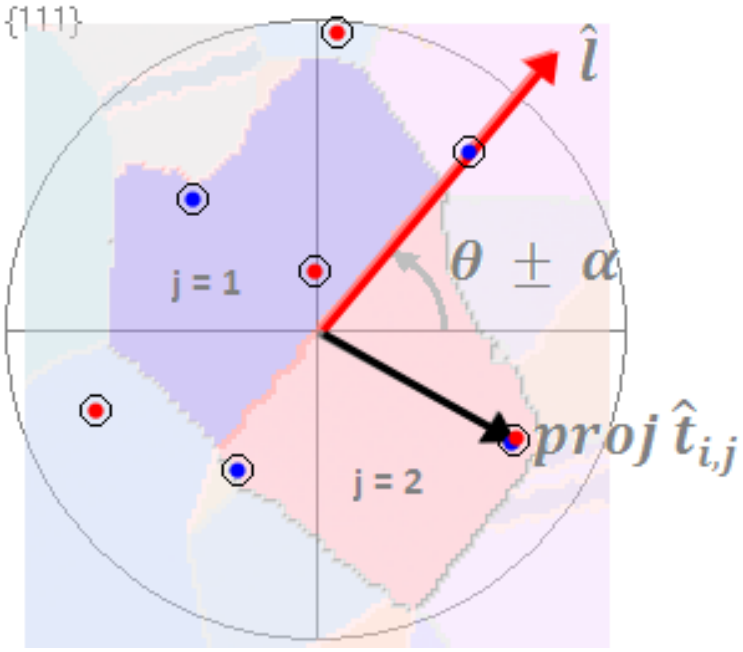


Figure 3.33 Pole figure representing the {111} interface normals of two grains, j = 1 & j = 2, adjacent to the Σ3 (red) boundary.

The error, defined here as α, in the angle, θ, occurs if the pixels that define the boundary trace are out of position. Figure 3.34 shows two overlaid EBSD maps, one with boundaries in red and the other blue, produced from the same EBSD data. The maps were subjected to the different HKL processing algorithms for assigning orientations to unindexed pixels, resulting in the pixels defining the boundary start and end points being offset. The boundary reconstruction procedure is applied to the same boundary in both maps producing a red line segment and a blue line segment. Due to the start and end points being offset by one pixel, the line segments are not coincident and produce different angles, θ.

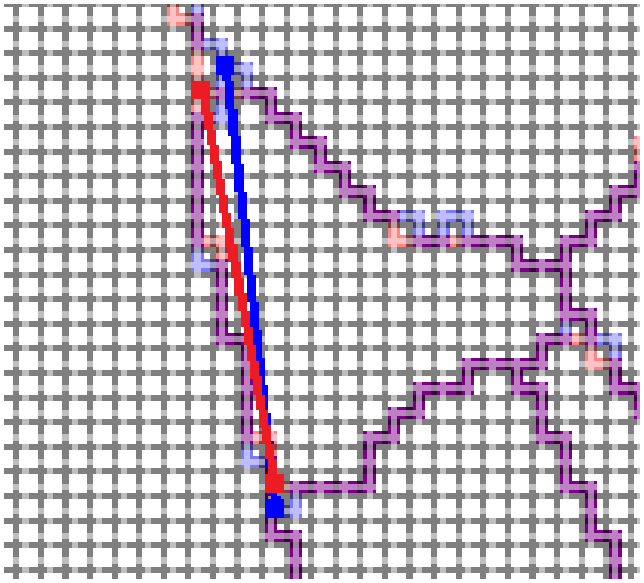


Figure 3.34 Two EBSD maps (red boundaries and blue boundaries) overlaid showing the offset of the triple points producing line segments of different orientations.

Figure 3.35 is a schematic of a line segment (red) of length, l, and angle, θ. Δ represents the mapping step size. The error, α, is given by Equation 3.14.

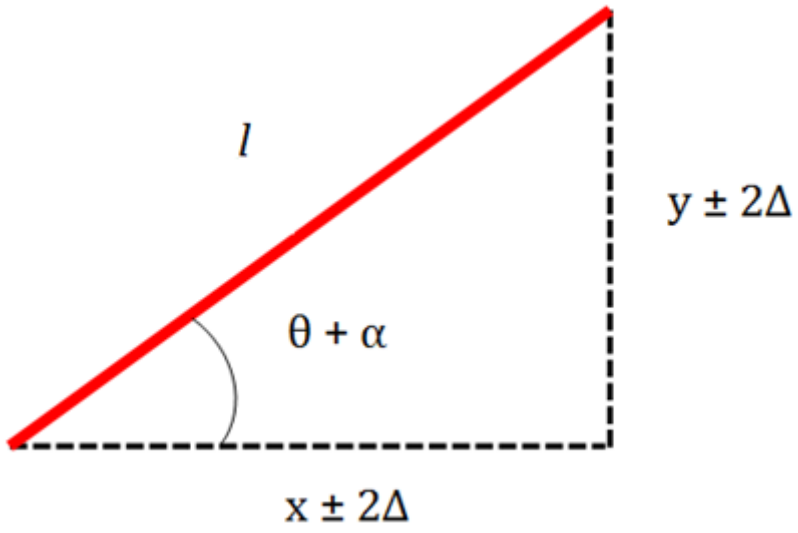


Figure 3.35 Schematic of grain boundary reconstructed line segment (red).

$$\alpha = max\left(\tan^{-1}\left(\frac{y \pm 2\Delta}{x \pm 2\Delta}\right)\right) - \theta$$

Equation 3.14

Figure 3.36 shows the angular error, α, for line segments of varying lengths represented in multiples of EBSD step size, *l/Δ,* at different orientations, θ. As expected, the error decreases with increasing segment length, and an error decrease is shown when the boundary trace is horizontal or perpendicular to the reference axis (θ = 0°).

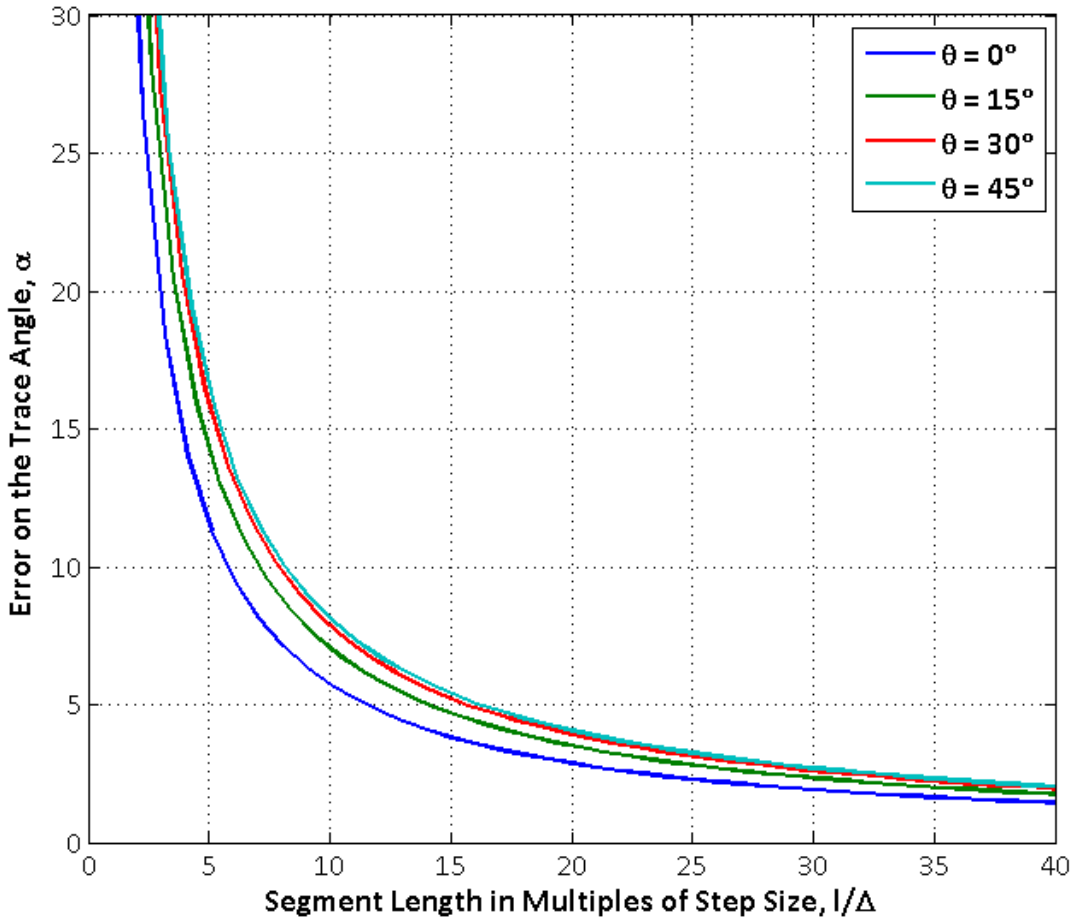


Figure 3.36 Error of the trace angle, α, for line segments of different length and orientation, θ.

To assess the model, 174,000 reconstructed Σ3 segments were analysed from 200 EBSD maps. The 200 EBSD maps were produced with step sizes ranging from 1 to 6µm, with average grain sizes between 20 and 180µm. Figure 3.37 shows the distribution of segment length identified as Σ3 boundaries in multiples of step size (l/Δ). After noise reduction approximately 55% of the Σ3 line segments were found to be shorter than 15Δ. The model, Figure 3.36, indicates that for segment lengths less than 15Δ, an error, α, greater than 5° may exist on the trace angle, θ.

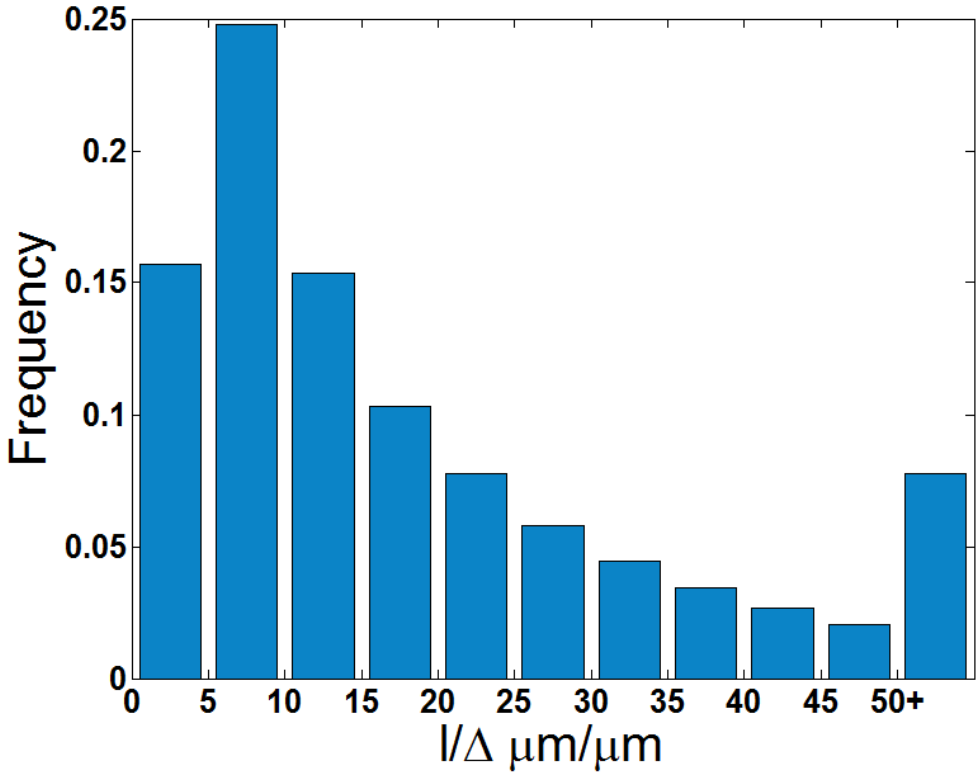


Figure 3.37 Distribution of the reconstructed Σ3 line segment lengths with length represented in multiples of step size, Δ.

Figure 3.38 shows the distribution of angles, α, between the trace of the Σ3 boundary segment and the trace of the {111} twinning plane. The Σ3 line segments are categorised by their length in multiples of step size, Δ. The result indicates that shorter Σ3 segment length does not correlate with larger angular errors, α, as predicted by the model, Figure 3.36. Table 3.7 shows the average value and standard deviation for the trace error, α. The average error, $\overline{\alpha}°$, for segments with lengths less than $\Delta 10$ was approximately 15% greater than for the larger segment lengths. While the difference in $\overline{\alpha}°$ may be the result of the reconstruction process, it is just as plausible that the short segment lengths represent the non-{111} boundary steps often observed along an extended length of coherent boundary. Table 3.7 shows the proportion of boundary segments identified as coherent using a value for α based on the model, Figure 3.36. The model identifies 83% of boundary segments with length less than 10Δ would be coherent, while 45% of segments with length greater than 50Δ would be defined as coherent. There is no reason why shorter boundary segments would be more likely to be coherent; in fact, it would make more sense for larger segments to be coherent due to the driving force towards the minimization of interfacial energy.

Randle [87] showed through serial sectioning that Σ3 boundaries with {111} interfaces (coherent twins) could have single-section trace angle, θ, deviations, α, up to 10°. Further studies [71, 131] suggested that any deviation, α, on the trace angle, θ, greater than 10° could be used to define a non-{111} interfaced Σ3 boundary. Using a constant α value of 10° [71, 131], Table 3.7 shows that approximately 75% of the boundary segments were coherent. This proportion is consistent with other studies where 70% of Σ3 boundaries were identified as coherent for alpha-brass and a nickel-based superalloy using trace analysis with a reconstruction tolerance of 10° [131]. Therefore, in the current study a constant value of 10° will be used for the trace error, α.

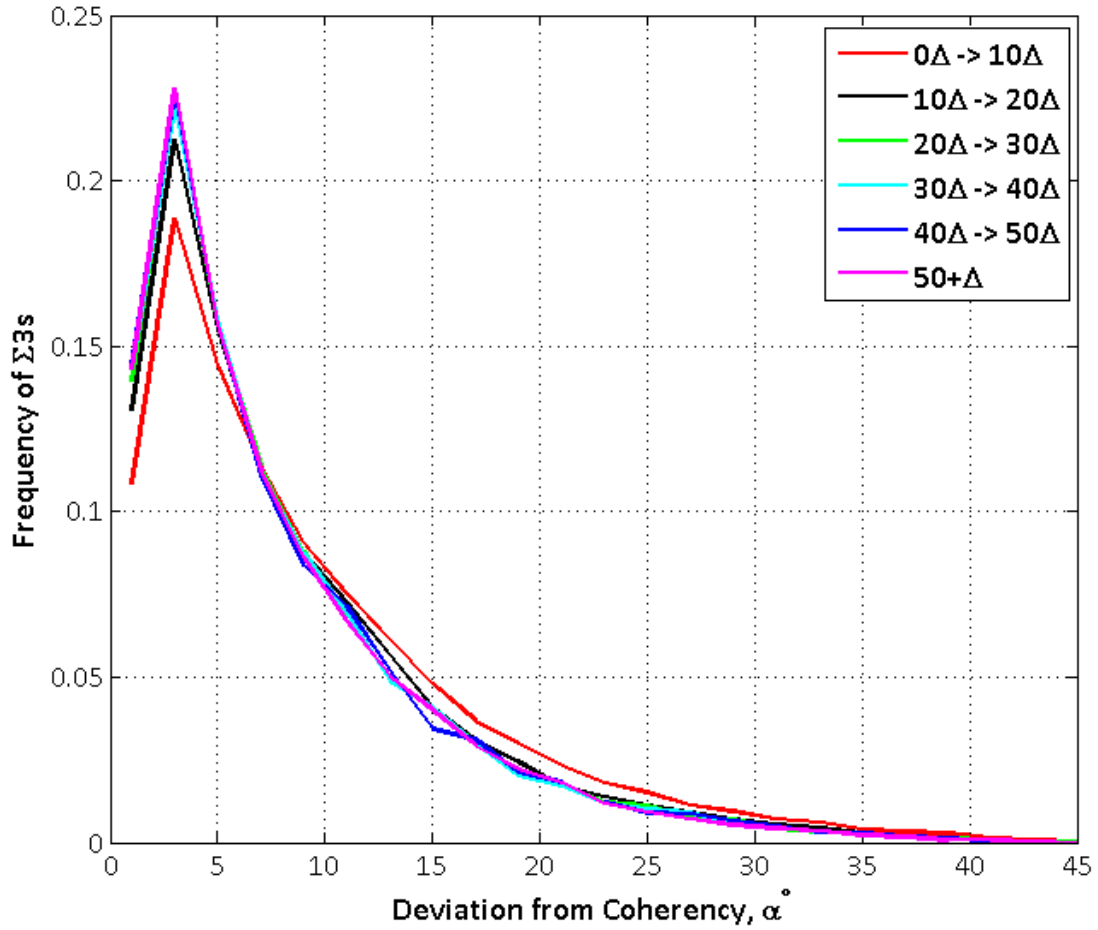


Figure 3.38 Distribution of the error, α, on the trace angle, θ, for the reconstructed Σ3 line segments categorised by their length in multiples of step size, Δ.

Table 3.7 Mean and standard deviation for the distributions shown in Figure 3.38. The proportion of coherent Σ3 segments identified using both the model and a constant value of 10° to define the trace error, α.

	Segment Length					
	Δ0 - Δ10	*Δ10 - Δ20*	*Δ20 - Δ30*	*Δ30 - Δ40*	*Δ40 - Δ50*	*Δ50+*
Average, $\bar{\alpha}$°	8.33	7.25	6.90	6.9	6.9	6.8
Standard Deviation	7.75	7.01	6.86	6.77	6.83	6.71
%Coherent - Model	83	78	63	58	53	45
%Coherent - 10°	68	74	76	76	76	76

3.5.8 Triple Junction Analysis

Triple junction analysis was performed using the method described by Kumar et al [119]. The method involves assigning each triple junction a category based on the number of $\Sigma 3^n$ ($n \leq 3$) boundaries which form that junction. Triple junctions are identified from the reconstructed EBSD maps with each grain boundary forming a triple junction defined as either non-$\Sigma 3^n$ or $\Sigma 3^n$. The triple junctions are categorised depending on whether they contained 0, 1, 2, or 3 $\Sigma 3^n$ boundaries.

An example of triple junction analysis is shown in Figure 3.39 for As Received Alloy 800H plate. The results indicate that the majority (87%) of triple junction contain either 0 or 1 $\Sigma 3^n$ boundary, while only 8% of triple junctions contains 3 $\Sigma 3^n$ boundaries.

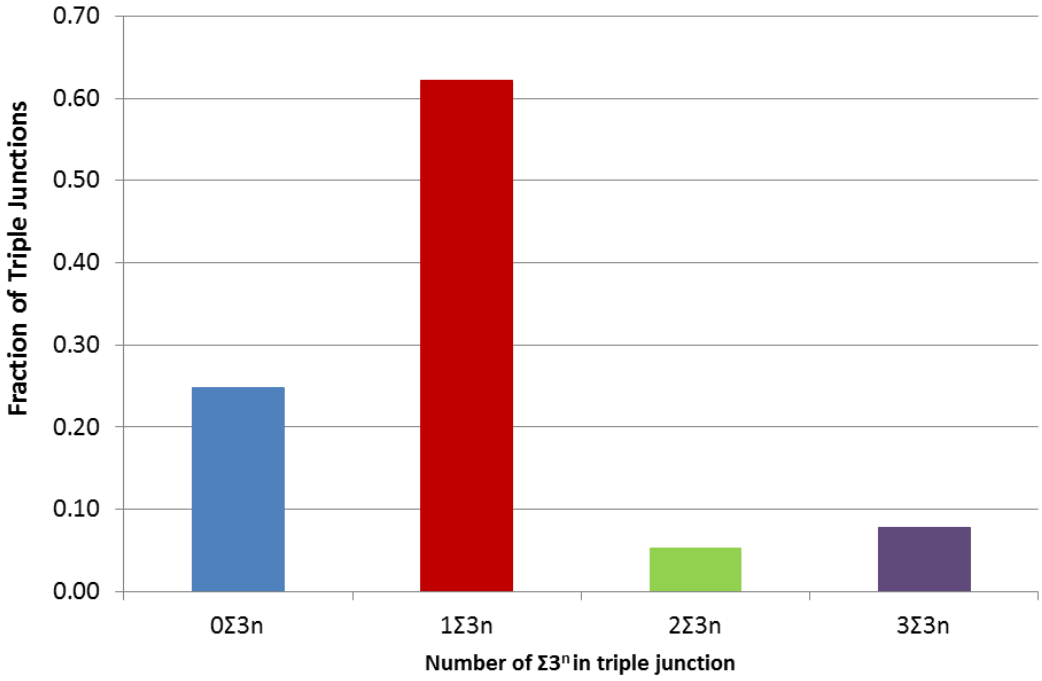


Figure 3.39 Triple junction analysis for As Received Alloy 800H plate.

3.6 Grain Size Measurement

3.6.1 Calculating Average Grain Size

The measurement of grain size plays an important role in microstructure-property relationship studies. As early as the year 1900 [132], a relationship between tensile strength and grain size had been established. It was soon recognised that there was a need for a standardized approach for the measurement of grain size in polycrystalline materials, hence the creation of the ASTM Committee E-4 in 1916. Committee E-4 was responsible for the development of a number of standards concerning metallography, including the first standard to deal principally with grain size measurement published in 1949, ASTM E79.

ASTM International produces a number of standards related to the measurement of grain size. These include:

1. ASTM E112 – Standard Test Methods for Determining Average Grain Size [3]
2. ASTM E2627 – Standard Practice for Determining Average Grain Size Using Electron Backscatter Diffraction (EBSD) in Fully Recrystallized Polycrystalline Materials [121]
3. ASTM E930 – Test Methods for Estimating the Largest Grain Observed in Metallographic Section (As Large As, ALA, Grain Size) [133]
4. ASTM E1181 – Test Methods for Characterizing Duplex Grain Sizes [134]
5. ASTM E1382 – Standard Test Methods for Determining Average Grain Size Using Semiautomatic and Automatic Image Analysis [135]

The test methods in ASTM E112 cover the measurement of average grain size, typically obtained from optical micrographs. Although the standard explains the measurement of grain size using several techniques, this study will focus primarily on the intercept procedure.

ASTM E112 defines a grain as “the area within the confines of the original (primary) boundary observed on the two-dimensional plane-of-polish. In materials containing twin boundaries, the twin boundaries are ignored, that is, the structure on either side of a twin boundary belongs to the grain”. The intercept procedure calculates the mean intercept distance between grain boundaries. The mean intercept distance is calculated using Equation 3.15:

$$\ell = \frac{L}{N^{\#}}$$

Equation 3.15

where ℓ is the mean intercept distance, L is the length of intercept line, and $N^{\#}$ is the number of boundary intercepts. The number of intercepts is counted from either straight lines placed at random orientations (Heyn Lineal Intercept Procedure), or circles positioned in random locations (Hilliard Single-Circle Procedure).

Figure 3.40 show the intercept procedure performed on Alloy 800H. Three 0.7mm diameter circles were drawn on an optical image and the number of boundary intercepts was counted. Twin boundaries were identified based on the morphological descriptions discussed previously, and, as per E112, were ignored. In total, 65 intercepts were counted for all three circles giving a mean intercept distance, ℓ, of 101µm.

ASTM E1382 provides instruction for measuring average grain size using automated image analysis. While grain size can still be measured in terms of the mean intercept distance, ℓ, image analysis also provides the ability to measure the area of the individual grain sections. From the individual grain area measurements, the average grain size can be calculated along with the grain size distribution, including identifying the smallest and largest grains.

Grain size can be expressed in a variety of ways; typically, the ASTM grain size number is the preferred method. The ASTM grain size equation is:

$$N = 2^{G-1}$$

Equation 3.16

where N is the number of grains per square inch at 100X magnification and G is the ASTM grain size number. The ASTM grain size number for mean intercept distance, ℓ, and the average grain area, $\bar{A}$, measurements are:

$$G = (-6.643856 \log(\ell)) - 3.288$$

Equation 3.17

$$G = (-3.3223 \log(\bar{A})) - 2.995$$

Equation 3.18

where ℓ is expressed in mm and $\bar{A}$ in mm^2. Table 3.8 gives the ASTM grain size number and the equivalent mean intercept distance, ℓ, average grain area, $\bar{A}$, and average grain diameter, $\bar{d}$, calculated using the equivalent circle diameter. The equivalent circle diameter (ECD) method assumes a circle of equivalent area describes the grain (equiaxed) and an equivalent diameter is calculated.

Table 3.8 ASTM grain size number and the equivalent mean intercept distance, ℓ, average grain area, $\bar{A}$, and average grain diameter, $\bar{d}$, calculated using the ECD method.

ASTM Grain Size - G	Mean Intercept Distance - ℓ (µm)	Average Grain Area - $\bar{A}$ (µm^2)	Average Grain Diameter - $\bar{d}$ (µm)
00	453	257960	573
0	320	128990	405
0.5	269	91213	341
1	226	64500	287
1.5	190	45610	241
2	160	32252	203
2.5	135	22807	170
3	113	16127	143
3.5	95	11404	121
4	80	8064	101
4.5	67	5703	85
5	57	4033	72
5.5	48	2852	60
6	40	2016	51

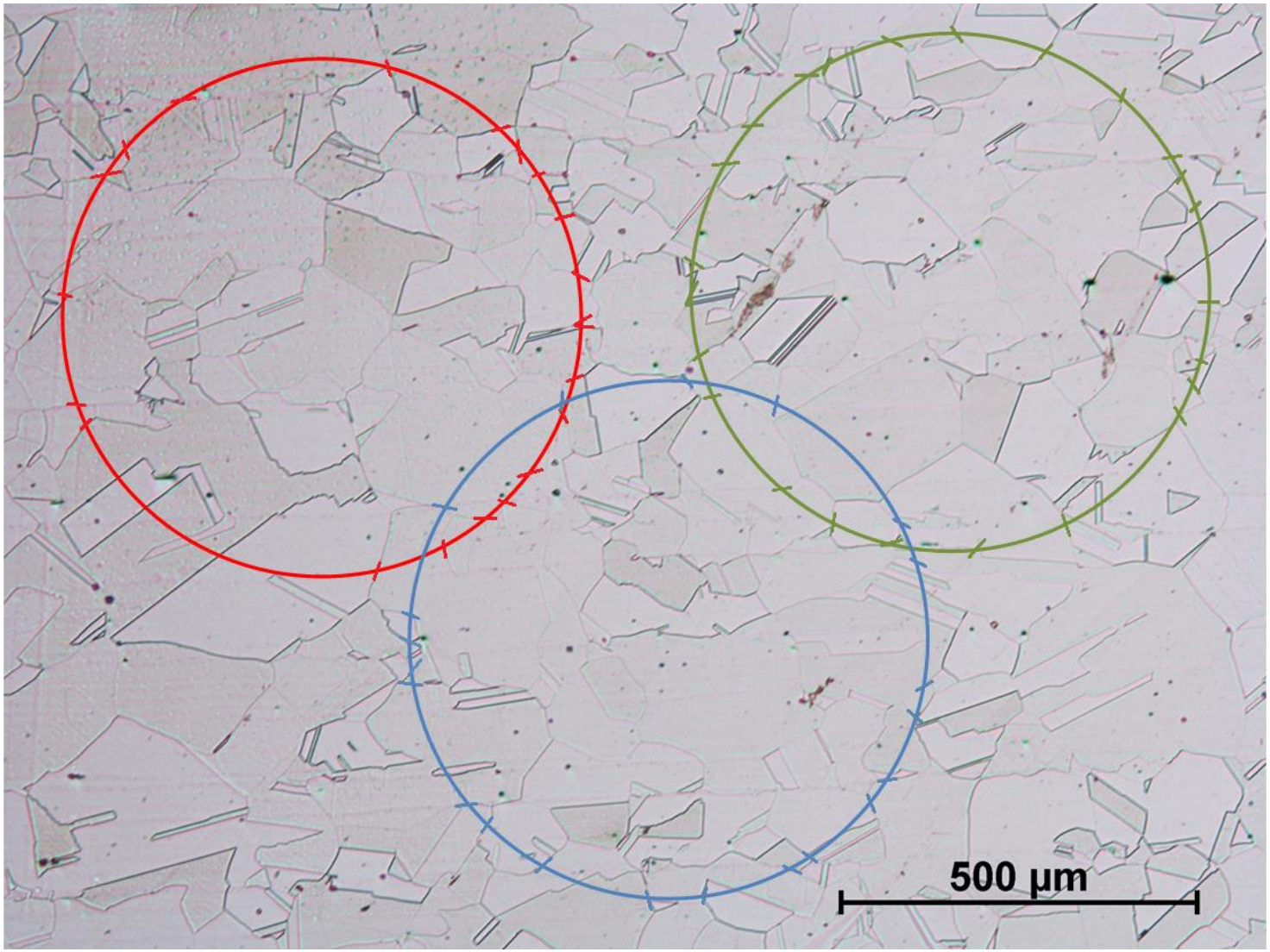


Figure 3.40 Circle intercept procedure performed on an optical micrograph of Alloy 800H. Mean intercept distance = 101µm (ASTM grain size = 3.3).

ASTM E2627 is a standard developed in response to the development of EBSD used to produce orientation maps. The EBSD mapping step size is selected based on the size of the average grain. Understandably some prior knowledge of the expected grain size is required. A step size is selected so the average grain contains approximately 500 EBSD measurements (pixels). For example, a sample with an average grain size of 100μm (ECD) will be mapped at a step size of 4μm.

Data clean-up is a routine procedure employed in EBSD mapping, whereby non-indexed or misindexed points are assimilated into surrounding grains. These poorly indexed points are generally a result of sample preparation, and also tend to be concentrated at grain boundaries. When the beam is positioned near a grain boundary, the diffraction volume will include the crystal lattices from both sides of the boundary. The EBSD pattern produced will be a composite of the individual patterns from the two grains and is difficult to solve. No map with less than 90% successful pixel identification rates should be used [121].

A grouping of points of similar orientations defines a grain. In the current study, a grain is defined as a grouping of points with misorientations less than 15°. Because EBSD is able to provide information related to grain boundary types, the option to omit certain boundaries arises. In the current study, coherent twin boundaries are ignored for grain size calculations; therefore, the areas adjacent to a coherent twin boundary are considered to belong to the same grain.

ASTM E2627 instructs users to eliminate all grains with pixel counts less than 100. The reason given for this number is historical compatibility with existing standards. ASTM E1382 outlines the procedure used to identify grains off a Semiautomatic Digitizing Tablet. The standard suggests that to ensure measurement accuracy, the smallest grain on a photomicrograph should be about 5mm in diameter. Wright [136] explains that assuming a measurement field of a 512 x 512 pixel array on a 225 x 225mm monitor, a 5mm diameter grain would be 11.4 pixels in diameter. Assuming equiaxed grains, then the equivalent area would be 102 pixels. ASTM E1382 also notes that for automatic image analysers with higher resolution, grains smaller than 5mm may be measured with reasonable precision. In the current study, areas represented by five pixels or less are removed from the EBSD maps. The removal of clusters between 1 and 5 pixels was validated through grain size measurements from 3D reconstructed serial sectioned Alloy 800H (Chapter 4).

While ASTM E112 explicitly states that twins should be omitted from grain size measurement, other than the morphological indicators discussed previously, there is no way to identify a twin boundary from an optical image. Grain size measurement performed on Figure 3.40 was repeated by mapping the same area with EBSD, identifying coherent twins using the algorithm described previously, and calculating the

grain size. Both the average grain diameter (equivalent circle diameter) and the mean linear intercept distance were calculated. The EBSD boundary maps, including and excluding twins, are shown in Figure 3.41. Table 3.9 displays the mean linear intercept distance for measurements performed on the optical micrographs, Figure 3.40, and the EBSD boundary maps, Figure 3.41. The average grain size diameter calculated from the EBSD map using the equivalent circle diameter is also provided.

The twin included intercept distance was 55µm for optical and 56µm for EBSD. These values indicate that the number of boundaries that were not successfully revealed through etching is approximately equal to the number of boundaries that were not identified through EBSD mapping. The twins excluded intercept distance was 101µm for optical and 77µm for EBSD respectively. The difference between the two measurements indicates that only approximately half of the coherent twin boundaries identified optically through their morphology were identified through EBSD and trace analysis. This result is consistent with the studies performed by Randle [82-84]. Randle showed through serial sectioning that approximately half of the Σ3 boundaries producing straight traces on a polished section had planar indices indicating a coherent twin. This result reinforces the idea that twin boundaries cannot be identified simply by analysing their 2D morphology.

Table 3.9 Grain size measurements, including and excluding twins, for Alloy 800H performed using optical micrograph and EBSD mapping.

	Optical Micrograph		**EBSD Boundary Map**	
	Twins Included	**Twins Excluded**	**Twins Included**	**Twins Excluded**
Mean Linear Intercept Distance	55µm	101µm	56µm	77µm
ASTM Grain Size	5.1	3.3	5.1	4.1
Average Grain Size Diameter (ECD)			46µm	58µm
ASTM Grain Size			6.2	5.6

For both twin included and twin excluded, the EBSD boundary map produced lower ASTM grain sizes for the ECD measurement compared to the mean linear intercept distance measurement. This disparity is because the ECD average grain size includes grain measurements from the entire grain size distribution,

while with the mean linear intercept method the probability of a randomly placed line intersecting the smallest grains is low. For example, 50% of the total sectioned area in the EBSD boundary map shown in Figure 3.41(B), is the result of largest 10% of the grains.

A major drawback for EBSD is that acquiring data tends to take significantly longer than with optical methods, although EBSD systems are becoming faster. For example, ASTM E2627 recommends a minimum of 500 grains be mapped with the average grain containing 500 EBSD measurements. This map would take just over four minutes to produce using the Hikari XP EBSD camera [81] indexing at 1000 points/second.

The average grain size, $\bar{d}$, is the arithmetic mean given by:

$$\bar{d} = \frac{1}{n}\sum_{i=0}^{n} d_i$$

Equation 3.19

where d_i is the equivalent circle diameter of a grain area from an EBSD map and *n* is the total number of grains.

The error for average grain size is calculated using the method provided in ASTM 2627 [121]:

$$\%Error = \frac{95\%CI}{\bar{A}} x100$$

Equation 3.20

where $\bar{A}$ is the average grain size area, and the 95% confidence intervals is:

$$95\%CI = \pm\frac{t \cdot S}{\sqrt{N}}$$

Equation 3.21

where N is the number of grains, t = 1.960 for $N \geq 500$, and the standard deviation, S, of the areas of individual grains, d_i, is:

$$S = \left[\frac{1}{N}\sum_{i=1}^{N}\left(d_i - \bar{d}\right)^2\right]^{\frac{1}{2}}$$

Equation 3.22

To describe the width of the grain size distribution, the coefficient of variation (CV) is used given by:

$$CV = \frac{S}{\bar{d}}$$

Equation 3.23

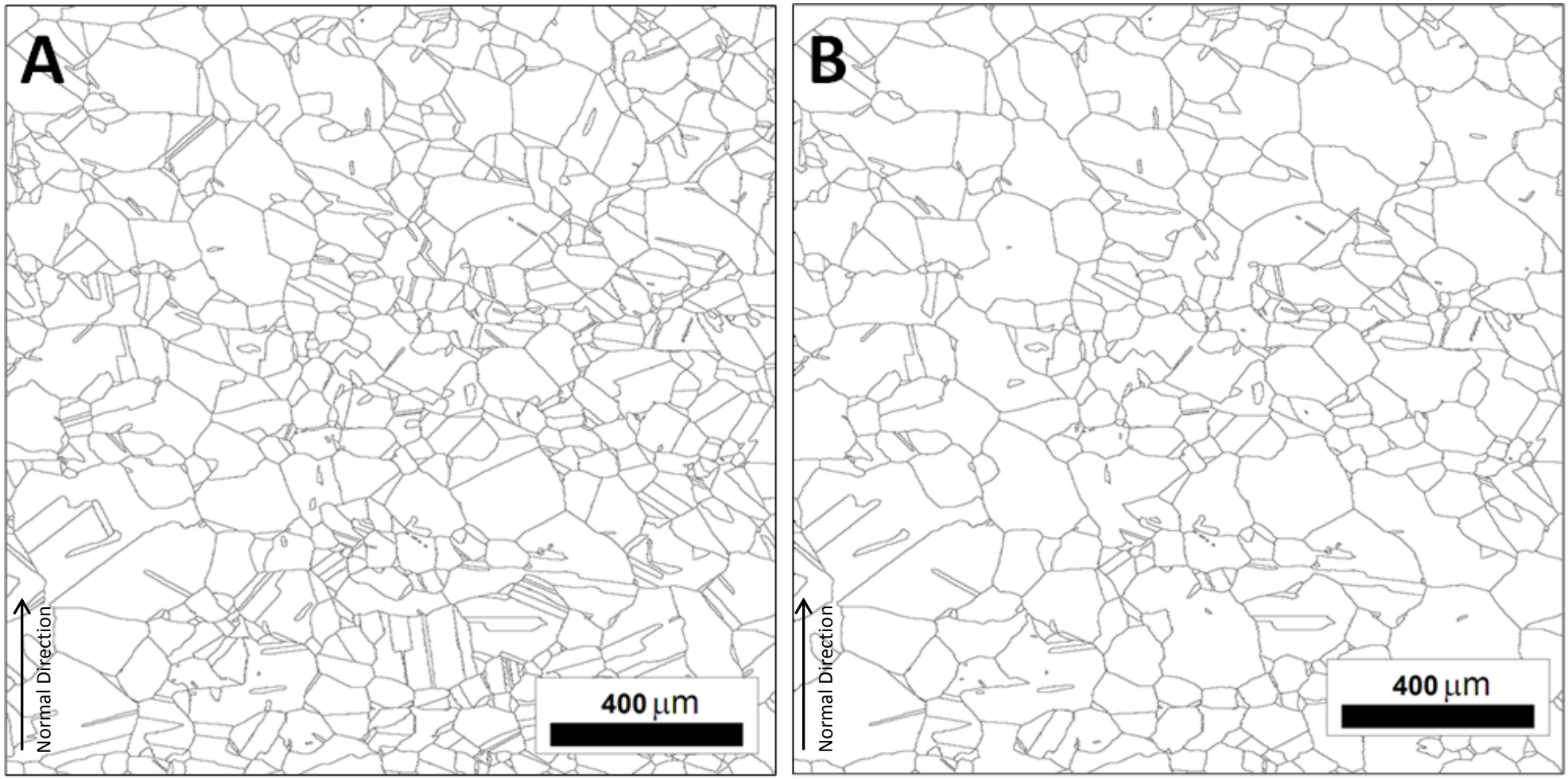


Figure 3.41 EBSD map from the same area shown in Figure 3.40 showing all boundaries (A), and the boundaries after twins have been removed (B).

3.6.2 Saltykov's Stereographic Correction

Stereographic corrections have been proposed for estimating a three dimensional grain size distribution from two dimensional measurements. The methods are typically based on the grain size measurement strategy, for example linear intercept [137, 138] or area [139].

Saltykov [139] proposed a method requiring the measurement of the two dimensional grain size distribution, $f(A)$, where A is the area of the sectioned grain. Saltykov's method assumes that grains are spherical in shape and the measured distribution can be represented by dividing the data into discrete size classes. The method is based on the probable diameters, *d*, which result from the intersection of a plane and a sphere. Figure 3.23 illustrates the ability of different sized sections (diameter, d_j) to exist from a single sized sphere arising from the random nature of a sectioning plane. For example, spheres of the largest class interval, D_1, can produce a two-dimensional area of any diameter, $d \leq D_1$, when sectioned.

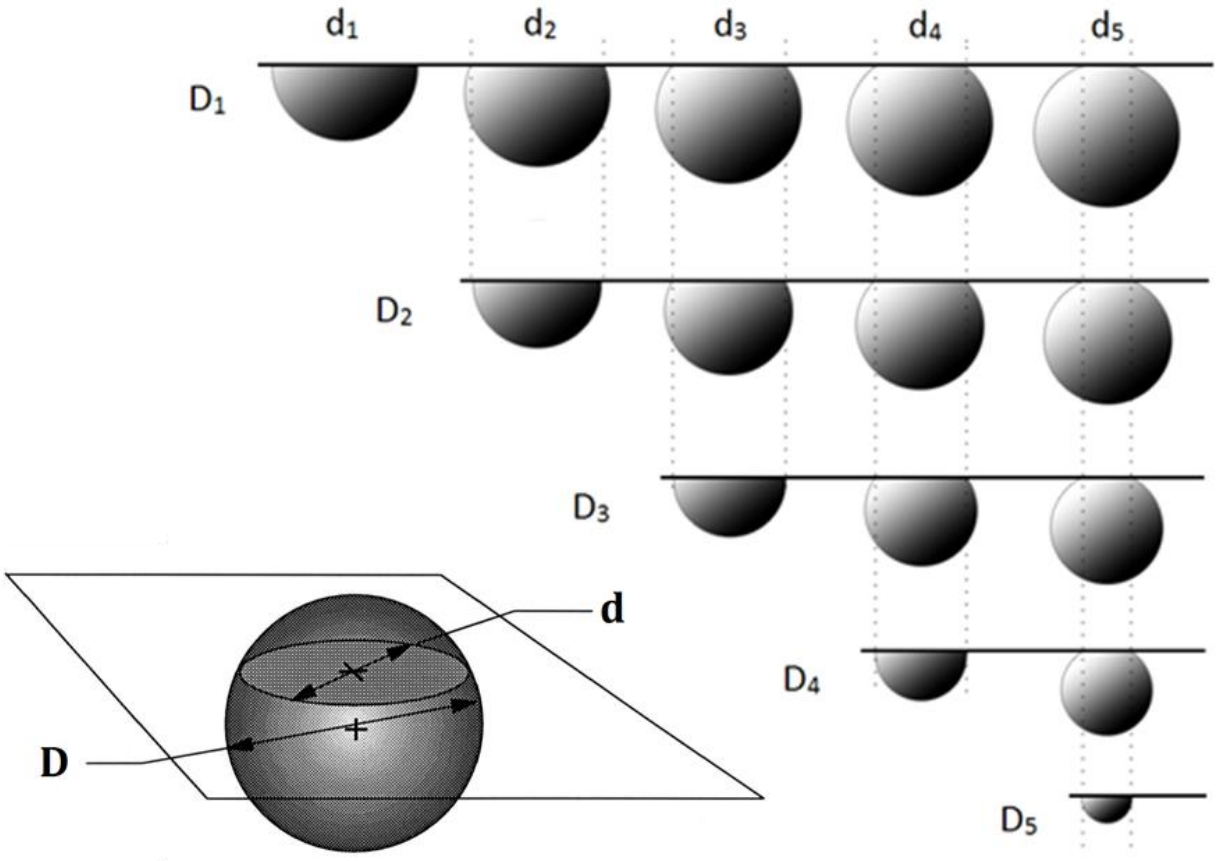


Figure 3.42 Schematic illustration of the contribution of spheres of diameters D_1 to D_5 to the total number of sections with diameters d_1 to d_5. (Figure adapted from [36]).

A grain section with a measured diameter, d, must be from a grain which has a diameter, D, that is greater than or equal to the measured one. If we consider the largest grain class size first, the true number of grains in this class size will be greater than the observed number, since some of the grain section diameters in the smaller class sizes are due to truncation of grains from the largest class size. The number of grains per unit volume which are estimated to be in the largest class size, N_V, is determined by applying a probability correction to the number of grains observed per unit area, N_A.

For all class sizes smaller than the largest, a further correction is applied. As well as correcting for the fact that grain sections in smaller class size are due to truncation of grains in the current class size (i.e. same correction as before), we must also subtract from the number of measured grain sections in the class size that actually belong to larger class sizes. The equation to calculate the true number of sections in each class size is given as:

$$(N_V)_j = \frac{1}{D_j}\left[1.641(N_A)_j - 0.4561(N_A)_{j-1} - 0.1162(N_A)_{j-2} - 0.0415(N_A)_{j-3} \ldots - 0.570x10^{-9}(N_A)_{j-29}\right]$$

Equation 3.24

where N_V is the corrected (true) number of grains for class interval j per unit volume, D_j is the grain size for the current class size and $(N_A)_j$ is the number of sectioned grains in class interval j. The class intervals are defined by the logarithmic scale factor $10^{-0.1}$. The list of 30 coefficients used in Equation 3.3 is given in Table 3.10.

Table 3.10 Saltykov coefficients based on size intervals related by $10^{-0.1}$ [140]

1.6461	0.4561	0.1162	$0.4149x10^{-1}$	$0.1727x10^{-1}$
$0.7795x10^{-2}$	$0.3684x10^{-2}$	$0.1790x10^{-2}$	$0.8841x10^{-3}$	$0.4414x10^{-3}$
$0.2218x10^{-3}$	$0.1119x10^{-3}$	$0.5665x10^{-4}$	$0.2872x10^{-4}$	$0.1457x10^{-4}$
$0.7401x10^{-5}$	$0.3761x10^{-5}$	$0.1911x10^{-5}$	$0.9716x10^{-6}$	$0.4939x10^{-6}$
$0.2511x10^{-6}$	$0.1277x10^{-6}$	$0.6493x10^{-7}$	$0.3302x10^{-7}$	$0.1679x10^{-7}$
$0.8537x10^{-8}$	$0.4341x10^{-8}$	$0.2207x10^{-8}$	$0.1122x10^{-8}$	$0.570x10^{-9}$

A source of error in Saltykov's correction is the assumption that the grain shape is spherical, since an array of spherical grains cannot fill space. Other methods have been proposed using more realistic grain shapes. For example, Takayama [141] proposed a method assuming all grains are tetrakaidecahedral in

shape, and therefore space-filling. Matsuura and Itoh [142] proposed an alternative method based on a range of grain shapes. They used twelve different types of equiaxed polyhedrons to represent a range of grain shapes.

Tucker et al [143] applied the Saltykov correction to a 3D data-set of Ni-based superalloy Inconel 100 to validate its application to 2D measurements. The results showed that, although the entire range of grain sizes within the distribution was not completely restored, the mean and upper tail was significantly improved. The simplicity of applying the Saltykov model to the current EBSD grain size measurements makes it the most appropriate method of correcting grain size distributions in the current study. The effectiveness of the Saltykov's correction is assessed in Chapter 4 when the 3D grain size measured from serial sectioned data is compared to 2D grain size measurements.

3.7 Summary

Chapter 3 detailed the material preparation techniques and methodologies used to identify coherent twins and obtain grain size measurements in Alloy 800H. Primary focus was on the processes involved in the effective collection of EBSD data, and the post processing required for analysing results.

The effect of EBSD mapping step size on the identification and quantification of boundary types, in particular Σ3s, was discussed extensively. It was shown that the total number of Σ3 boundaries and their fragmented appearance was strongly dependent on the EBSD mapping step size. While mapping anomalies such as the fragmentation of Σ3 boundaries are unavoidable, their effects can be limited by employing a relative length fraction to quantify boundary types. For samples with grain sizes ranging from 20 to 250µm, step sizes of 1 to 5µm were selected.

An algorithm developed for the analysis of EBSD maps (Appendix B) was discussed. Small pixel clusters (1-5µm) representing false grains are removed from the EBSD maps prior to the reconstruction of boundaries with line segments. Using the orientation of the line segments and the crystal orientation of the adjacent grains, provides four out of the five degrees of freedom necessary to define the boundary interface. Trace analysis was subsequently used to determine which Σ3 boundaries could be considered coherent twins.

Grain size measurement methodologies detailed in ASTM E112 and E2627 were compared. The mean linear intercept method was shown to calculate ASTM grain size numbers larger than those calculated through average grain area. This measurement discrepancy is due to the over sampling of larger grains when placing intercept lines. EBSD and trace analysis was also shown to identify fewer twin boundaries compared with optical microscopy and employing morphological descriptions. This result is consistent with the studies performed by Randle [82-84]. Randle showed through serial sectioning that only approximately half of the Σ3 boundaries producing straight traces on a polished section had planar indices indicating a coherent twin. This result reinforces the idea that twin boundaries cannot be identified simply by analysing their 2D morphology.

Saltykov's stereographic correction for estimating a 3D grain size distribution from 2D measurements was discussed. The method is based on the probable diameters, *d*, which result from the intersection of a plane and a sphere. Tucker et al [143] applied the Saltykov correction to a 3D data-set of Ni-based superalloy Inconel 100 to validate its application to 2D measurements. The results showed that, although the entire range of grain sizes within the distribution is not completely restored, the mean and upper tail was significantly improved.

The current study will apply the Saltykov correction to grain area measurements made from EBSD orientation maps. Twin boundaries will be identified and removed from the EBSD mapped microstructure using trace analysis prior to grain size measurement. The grain size measurement methodologies described in this chapter will be assessed in Chapter 4 by comparing the grain size measured from 3D reconstructed grain volumes with that measured from 2D sections.

Chapter 4 3D Reconstruction of Twinned Austenite Grains

4.1 Introduction

Chapter 4 details the serial sectioning and three-dimensional (3D) reconstruction of Alloy 800H. The 3D morphology of crystal volumes reveals deficiencies in the traditional system of classifying geometry from two-dimensional (2D) sections. The formation of boundary facets and the resulting implications for Σ3 coherency is discussed. A proposed mechanism for the formation of facetted twin boundaries is presented. Finally, the 3D reconstruction data is used to validate trace analysis for the identification of twin boundaries, and stereographic corrections in the measurement of grain size statistics.

4.2 Method

4.2.1 Sample

The sample selected for serial sectioning was prepared from the AR plate material. While thermo-mechanical processing made it possible to prepare a sample with a microstructure favourable to the technical challenges of serial sectioning and reconstruction, e.g. smaller grain size or simple Σ3 geometries (i.e. increased RHGB connectivity), it was seen as more valuable to the study to analyse a microstructure typical of the As-Received and in service condition.

4.2.2 Serial Sectioning

Microhardness indents were used as fiducial markings, serving three purposes:

1. To define the periphery of the area of interest.
2. To align polished sections.
3. To calculate the depth of material removed.

After every fifth section, pairs of microhardness indents were placed in one of four locations around the periphery of the 1.6mm x 1.6mm area of interest. Using four different indent locations allowed for the placement of new indents while previous indents were still visible.

The depth of material removed was calculated based on a 7:1 relationship between the distance across the Vickers indenter diamond, Figure 4.1, and the depth of material removed. Sectioning was performed using a MiniMet® automated polisher and the sample etched using glyceregia. The sectioning process removed material to give a z-resolution of 2 ± 0.1µm per section.

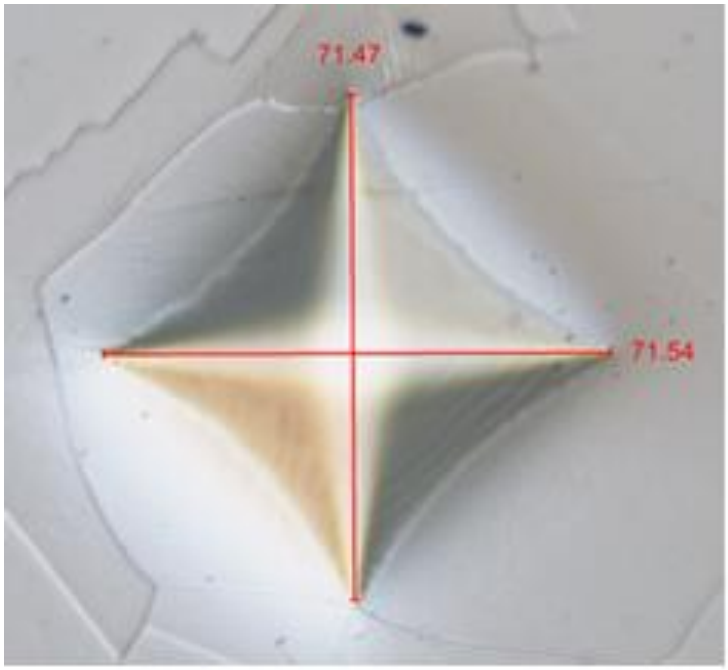


Figure 4.1 Measurement of the distance across the diamond indent.

Optical microscopy was used to locate boundary position. Each section is a montage of 143 images (11 wide x 13 high) captured at 500x magnification and manually aligned in Photoshop CS5, Figure 4.2. The sections have a resolution of 0.17µm per pixel. The final stack, also compiled in Photoshop, comprised 85 optical serial sections aligned manually using the microhardness indents.

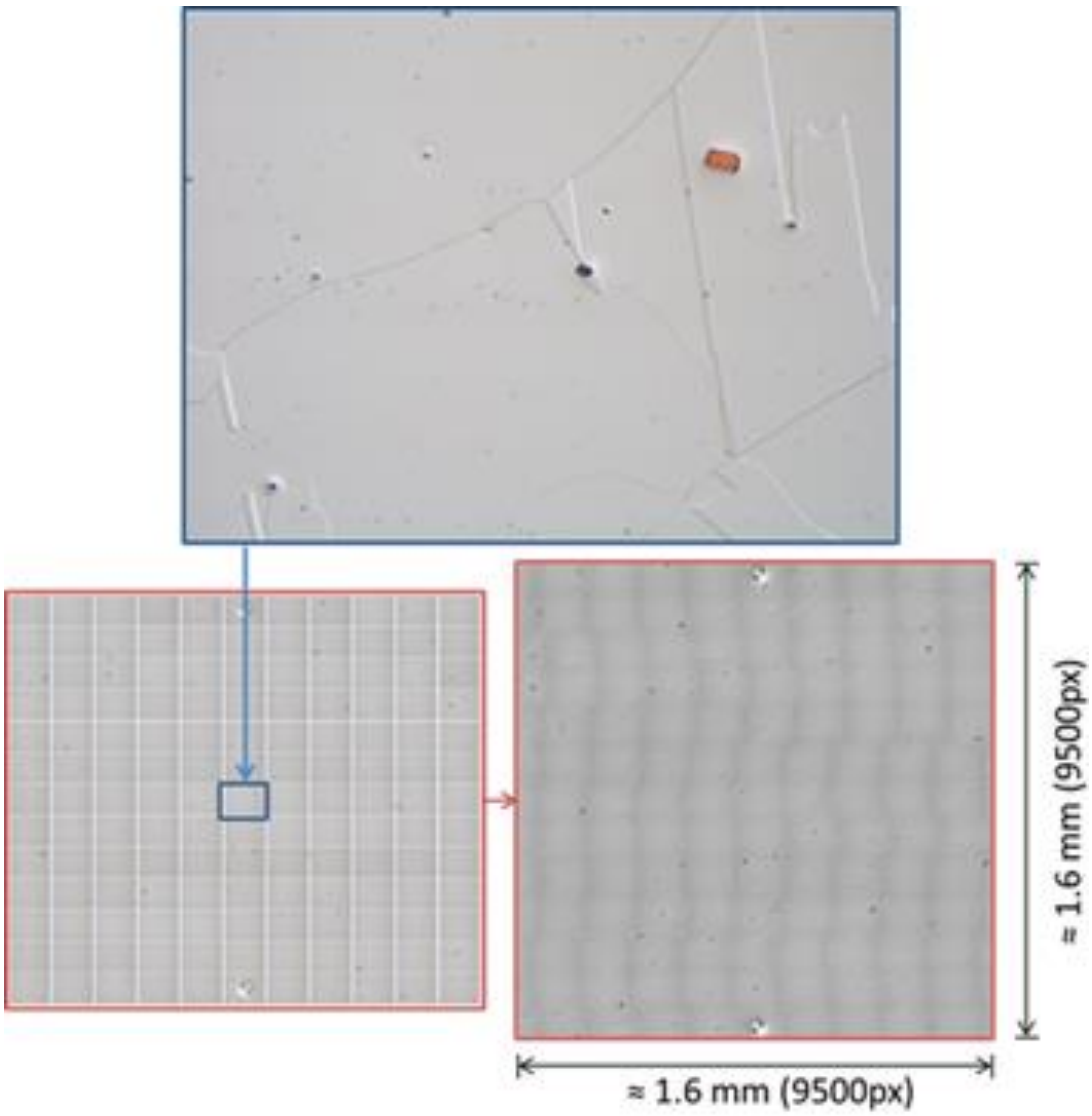


Figure 4.2 Montage of 143 optical images producing a single section.

EBSD maps with a step size of 1.5µm were produced every fifth section (every 10µm) to provide crystallographic orientation information. Average orientations from adjacent sections of the same volume were calculated, and a maximum misorientation between sections was found to be less than 1°, a value similar to other EBSD/serial sectioning studies [44].

4.2.3 Stack Segmentation, Reconstruction, and 3D Grain Size Measurement from EBSD Serial Sections

The EBSD maps were aligned with the optical images and cropped producing a sectioned volume of 1000µmx1000µmx160µm. Sectioned areas belonging to the same grain appearing on adjacent EBSD mapped serial sections were identified and assigned a unique colour. An example of a segmented EBSD map is shown in Figure 4.3 for serial section number 35.

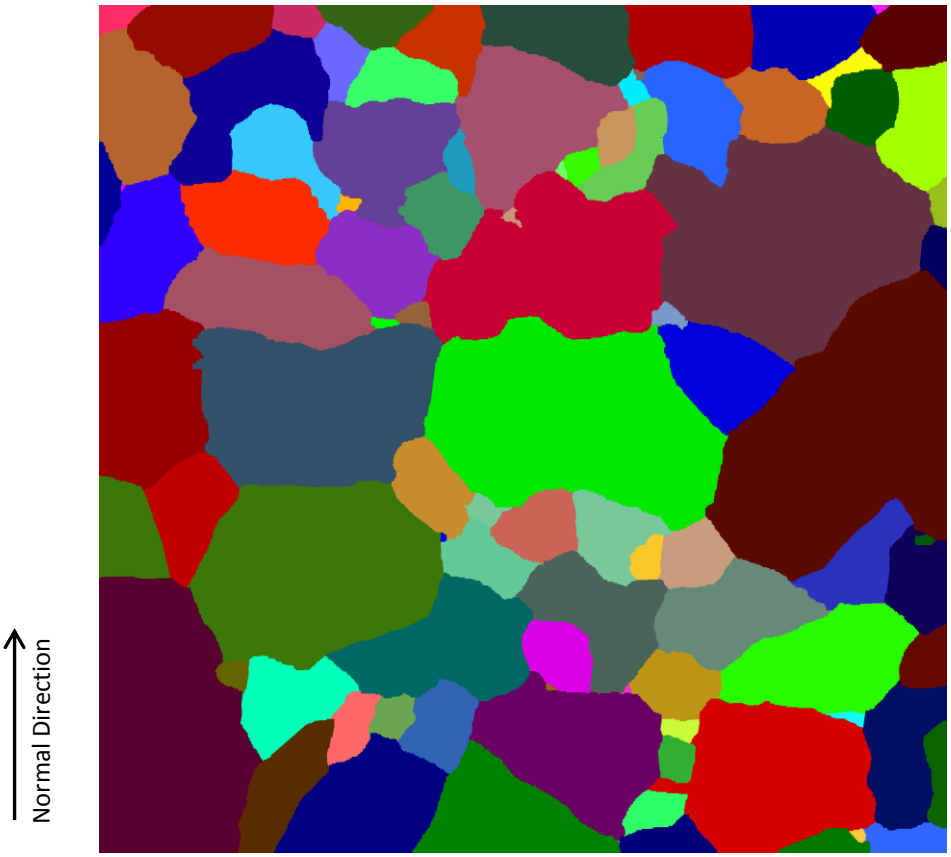


Figure 4.3 EBSD map of serial section number 35 with grains identified and segmented by assigning a unique colour.

3D reconstruction of the segmented EBSD serial sections was performed using IDL™ scripts (Appendix C) written by Dave Rowenhorst at the US Naval Research Laboratory (NRL). The IDL algorithm uses the aligned stack of serial sections to produce a 3D reconstruction, Figure 4.4(a-b), consisting of meshed voxels with x-y dimensions equal to 1.5µm (EBSD section mapping resolution) and z dimension equal to the distance between the sections (10µm). The voxelated surfaces of the grain volumes are refined (smoothed) using the inbuilt MESH_SMOOTH function (Laplacian smoothing) available in IDL (V8.2), Figure 4.4(c-d).

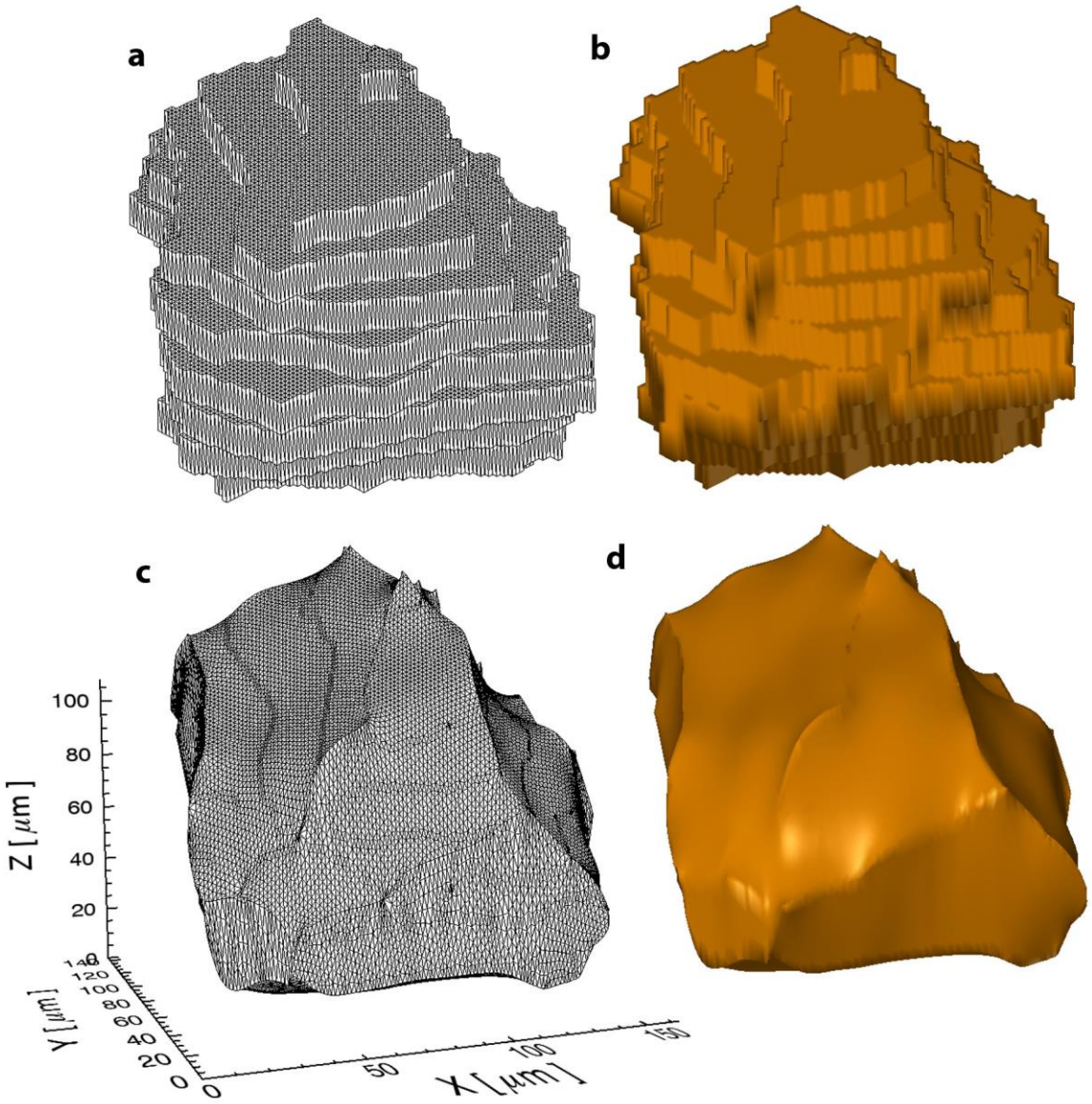


Figure 4.4 3D reconstruction of a segmented grain from serial sections. (a) Voxelated mesh. (b) Voxelated surface. (c) Smoothed mesh. (d) Smoothed Surface.

Figure 4.5 shows the 3D reconstruction of the EBSD maps producing a total of 170 grains. Only 28 grains out of the 170 grains reconstructed were complete, i.e. not truncated by the boundary faces of the serial sectioned volume. The z-axis is the rolling direction, y-axis the normal direction, and the x-axis if the transverse direction for the as-received plate material.

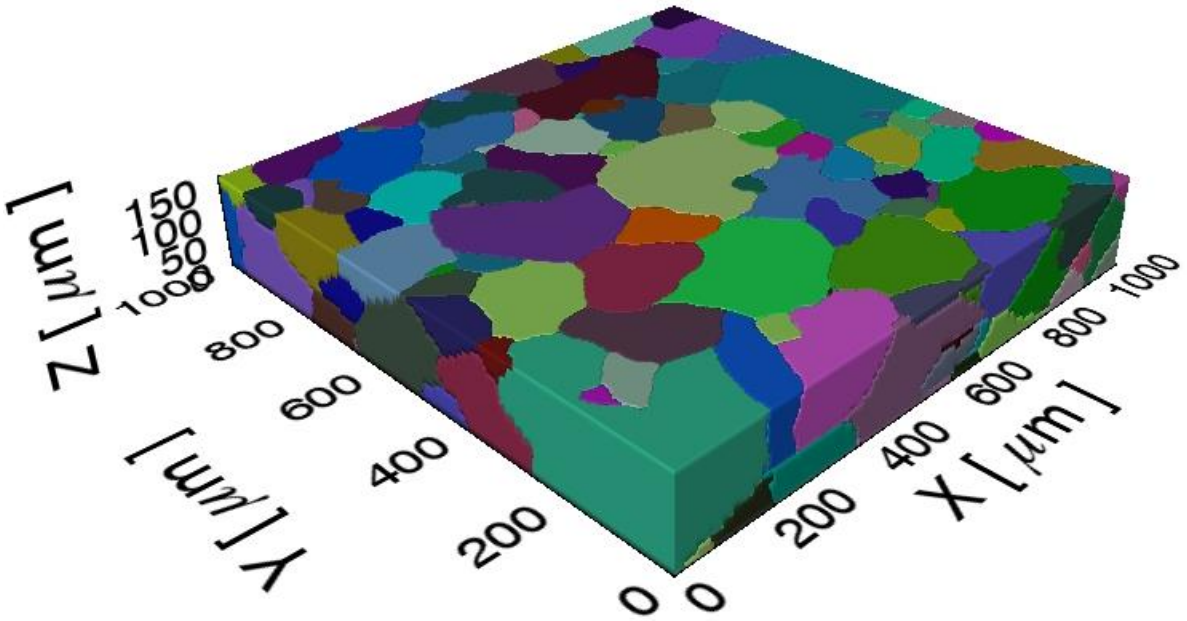


Figure 4.5 Reconstructed volume produced from segmented EBSD serial sections.

Figure 4.6 shows the distribution of the equivalent sphere diameters (ESD) for the 28 fully reconstructed grains. The average 3D grain size equals 55.5µm, given by:

$$\overline{d_{ESD}} = \frac{\sum_{N_g} d_{ESD}}{N_g}$$

Equation 4.1

where N_g is the number of grains and d_{ESD} is the ESD of the grain's reconstructed volume given by:

$$d_{ESD} = \sqrt[3]{\frac{6\,V_{grain}}{\pi}}$$

Equation 4.2

The volume of a 3D reconstructed grain, V_{grain}, is given by:

$$V_{grain} = V_{voxel}\,N_v$$

Equation 4.3

where N_V is the number of voxels in the grain. Each voxel in 3D space has a volume, V_{voxel}, given by:

$$V_{voxel} = \varepsilon\delta^2$$

Equation 4.4

where ε is the section thickness and δ is the x-y resolution.

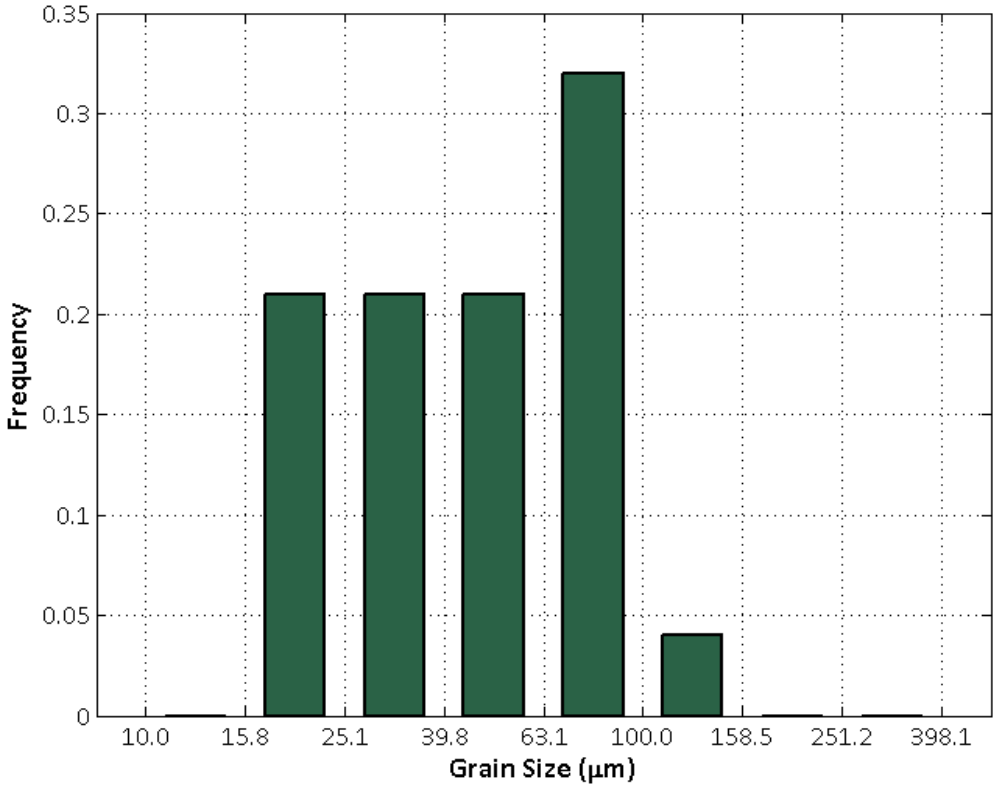


Figure 4.6 Grain ESD distribution of the 28 fully reconstructed grains.

Figure 4.6 does not show an upper tail or lower tail as would be expected for log-normal grain size distribution measurements. The absence of an upper tail is due to the cropped, sectioned volume not being sufficiently sized to provide a full reconstruction and volume measurement of the largest grains within the microstructure. The absence of small grains (less than 20µm) is due to the spacing between EBSD serial sections providing a lower limit to the size of grain that can appear over two adjacent sections.

An alternative measure of 3D grain size is to assume that the equivalent circle diameter (ECD) of the largest 2D grain section is comparable to the ESD of the reconstructed grain. This method assumes that grains are roughly equiaxed, and the largest section of the grain volume exists within the reconstructed volume. Figure 4.7 is an example of this process for a grain with a measured d_{ESD} of 120.2µm, and a d_{ECD} of 119.2µm for the largest section.

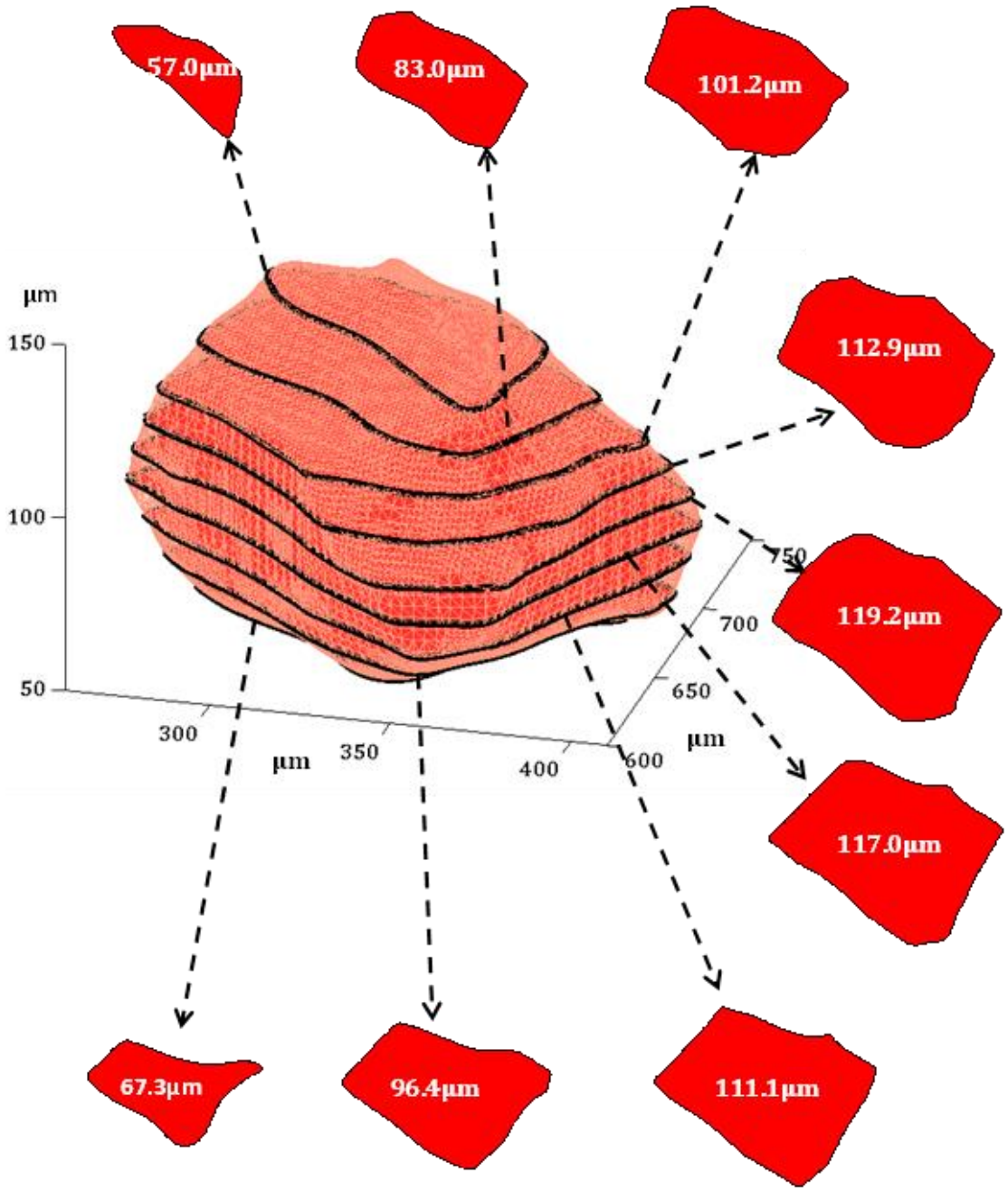


Figure 4.7 Example of the grain size measurements from a series of sections through a grain volume.

Figure 4.8 shows a measured d_{ECD} plotted against the measured d_{ESD} for the 28 fully reconstructed grains. The straight line fit produces an R^2-value of 0.98 and a slope of 1.04. The result indicates that the d_{ECD} value of the largest grain section is comparable to the measured d_{ESD} of the reconstructed grain.

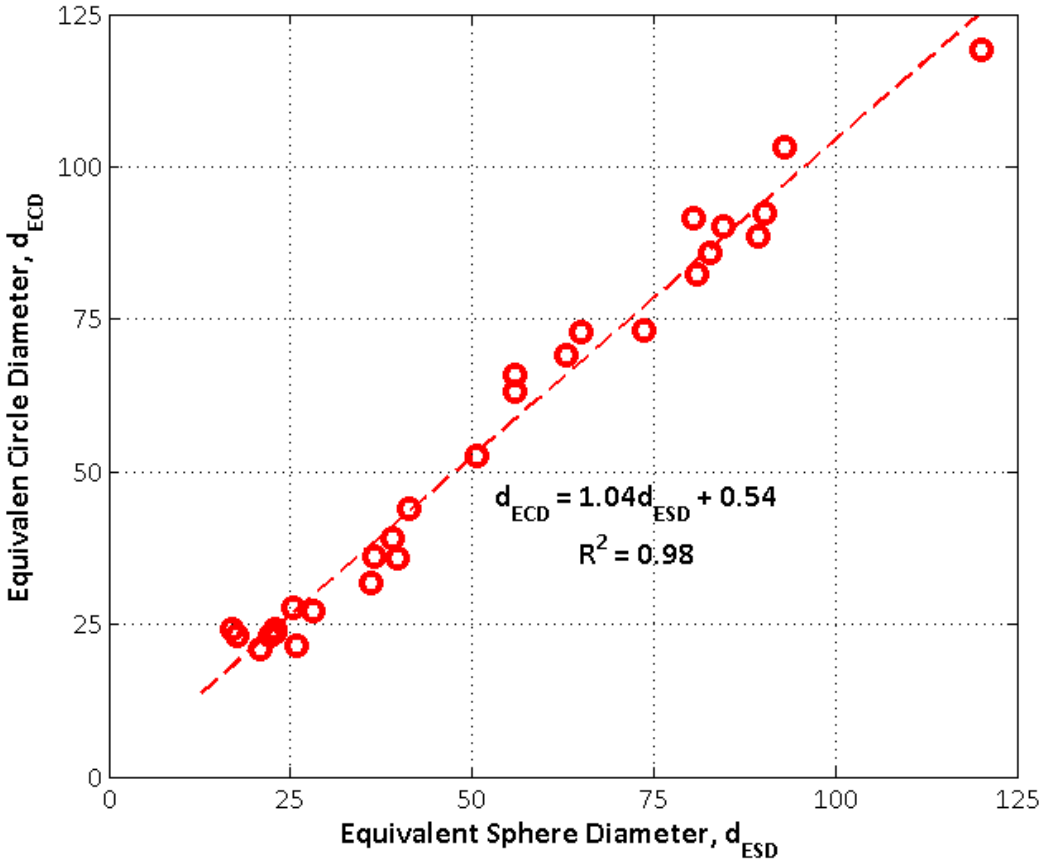


Figure 4.8 Comparing the d_{ESD} against the d_{ECD} for the 28 fully reconstructed grains.

The advantage of using the d_{ECD} value to describe the 3D grain size is that it allows for the smaller and larger truncated grains to be measured and included in the distribution. Because this method assumes that the largest 2D section exists within the volume, truncated grains with their largest section measured at a surface of the reconstructed volume were omitted.

From the 100 grains measured, 20 were identified from a single serial section (i.e. small grains), 28 were complete reconstructions, and 52 were truncated reconstructions. The smallest grain measured was 5.1µm and the largest grain measured was 284.2µm.

4.2.4 Reconstruction of Crystal Volume Clusters and Interface Measurement

Clusters of Σ3 crystal volumes (volumes interfaced by at least one Σ3 boundaries) were located in the stack of optical sections, Figure 4.9(a). Crystal boundaries were drawn manually using Photoshop, Figure 4.9(b), and, as before, each volume was assigned a unique identifying colour for the purpose of reconstruction, Figure 4.9(c). The 3D reconstructions performed using the IDL™ scripts, Figure 4.10 .

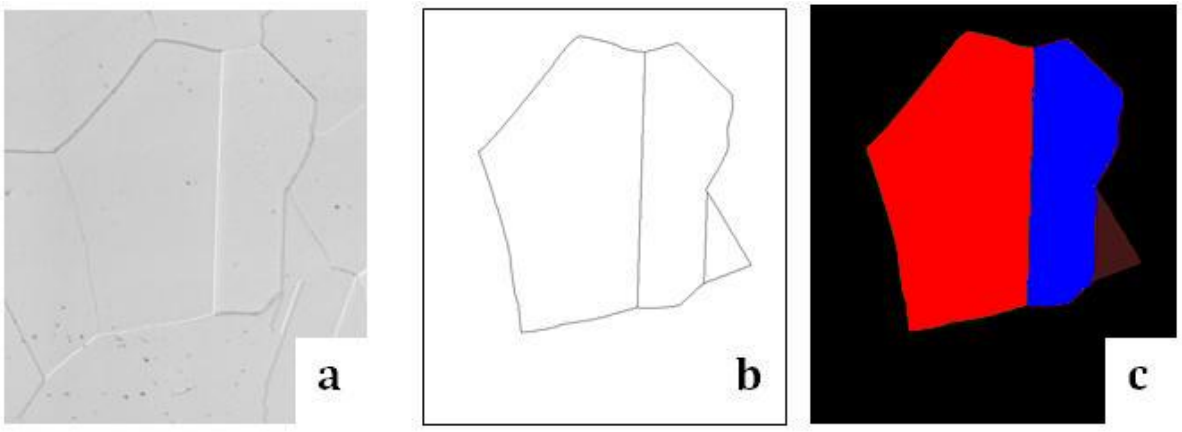


Figure 4.9 Crystal volume segmentation process: (a) identify clusters of crystal volumes from optical stack, (b) boundaries drawn manually in Photoshop and (c) crystal volumes assigned unique identifying colour.

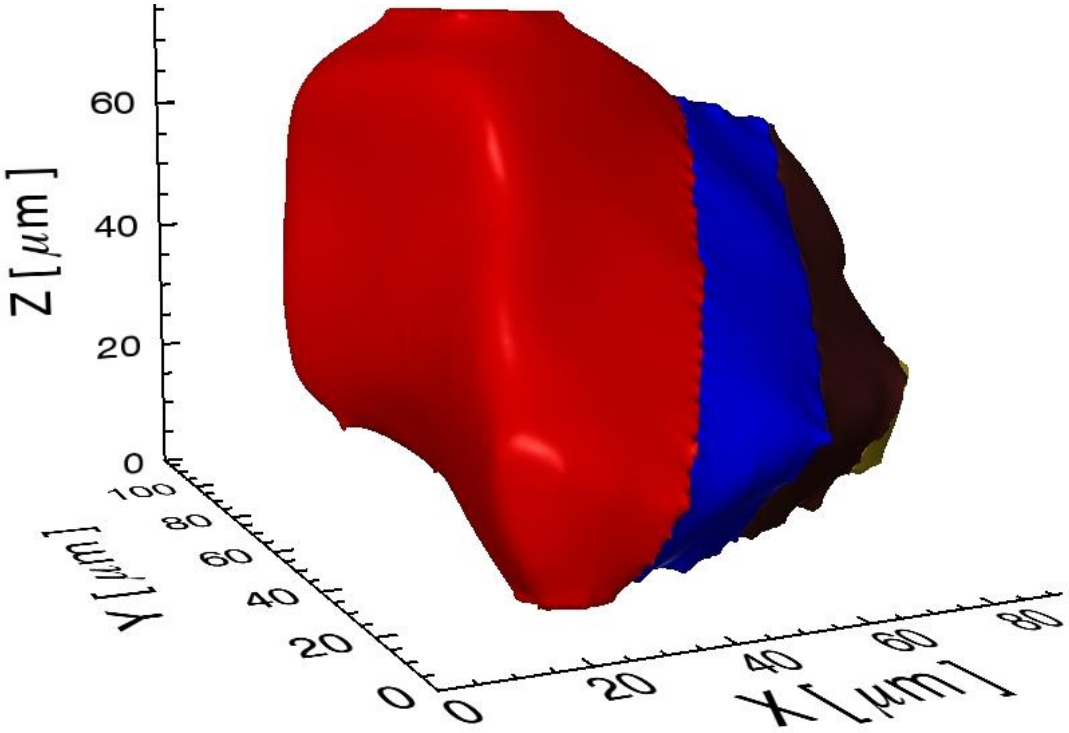


Figure 4.10 Reconstructed Σ3 cluster.

The use of the optical image stack of serial sections rather than the EBSD section stack allows for a five times increase in resolution in the z-direction (2µm between sections) and a nine times increase in the x and y directions (0.17µm pixels). The increase in resolution results in a more accurate reconstruction of the boundary interface planes permitting the measurement of interface orientation.

Figure 4.11(a) shows the identification of the triple point locations (green points) across adjacent serial sections formed by the intersection of three crystal volumes. Connecting triple points, Figure 4.11(b), forms a triple line (green line) resulting in an interface edge shared between the blue and red crystal volumes. The reconstruction of triple lines is repeated for all volume intersections until the boundary interface is completely bound by an interface edge. Figure 4.12 shows the 3D reconstruction of the blue crystal with an area of surface bound (green line) by an interface edge defining the Σ3 interface shared with the red crystal.

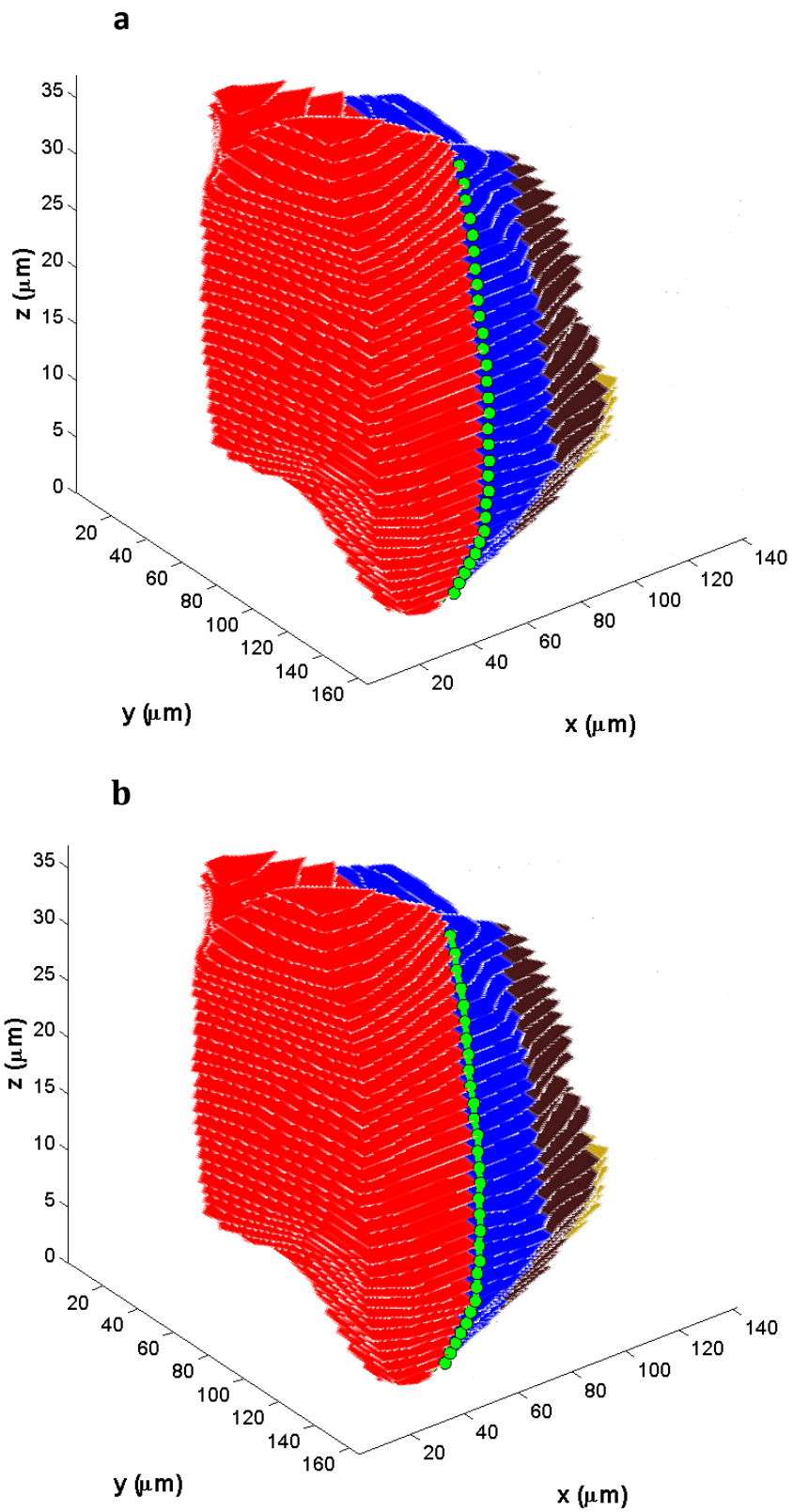


Figure 4.11 Locating (a) and connecting (b) triple points across multiple serial sections forming a shared interface edge.

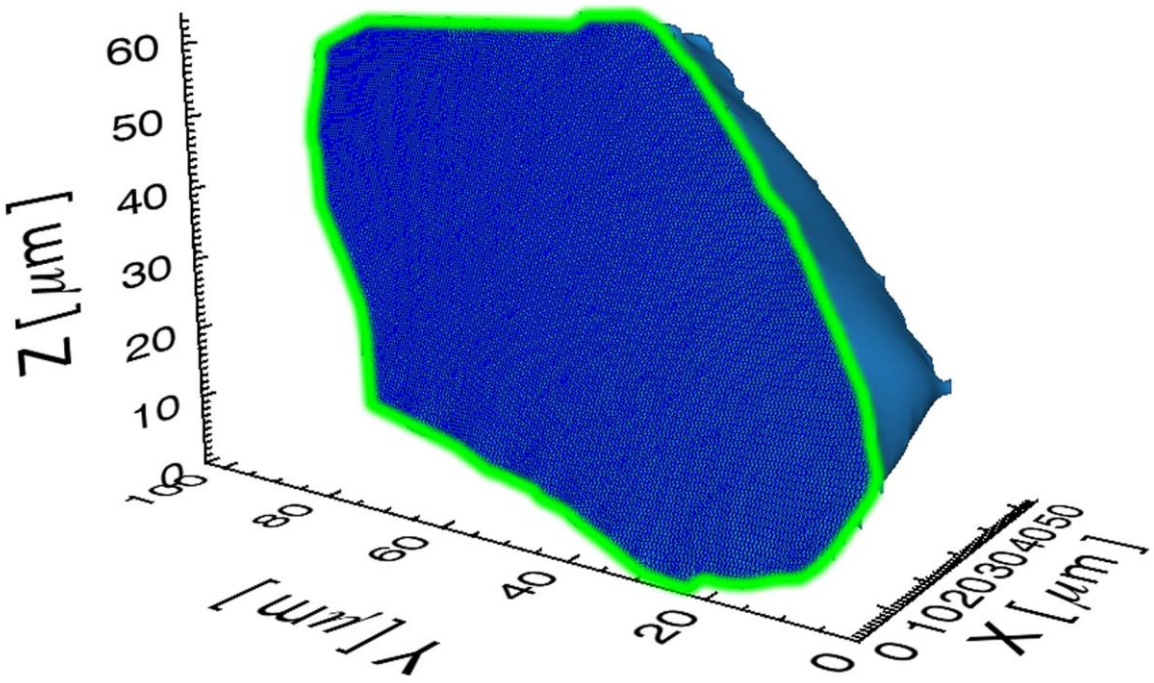


Figure 4.12 Crystal volume with an interface identified with a green outline.

The smoothed triangle mesh defining the interface surface, coupled with the EBSD orientation measurements, are used to identify the interface planes for the Σ3 boundaries. The normal vector to each triangular mesh element is compared to the <111> directions (coherent twin plane normal) in the adjacent crystal volumes. Thresholds (angle between mesh element normal vector and twin normal) of 6° [144] and 15° [145] have been employed in previous serial sectioning studies to account for section misalignment and segmentation errors. For the current study a threshold of 10° was used.

Figure 4.13 shows a point cloud on a unit sphere produced by the normal vectors of the 20,104 triangle elements that define the interface between the blue and red crystal volumes. The twin plane normal (normal vector to the {111} twinning plane) calculated from the orientation of the two adjacent crystals are shown along with circles encompassing regions representing the 10° threshold misorientation. The interface normals positioned within the region where the two circles overlap are thus considered coherent boundary elements. For the interface shown in Figure 4.12, 91% of the interface mesh elements are coherent. (See Figure 4.16 for the complete reconstructed volumes).

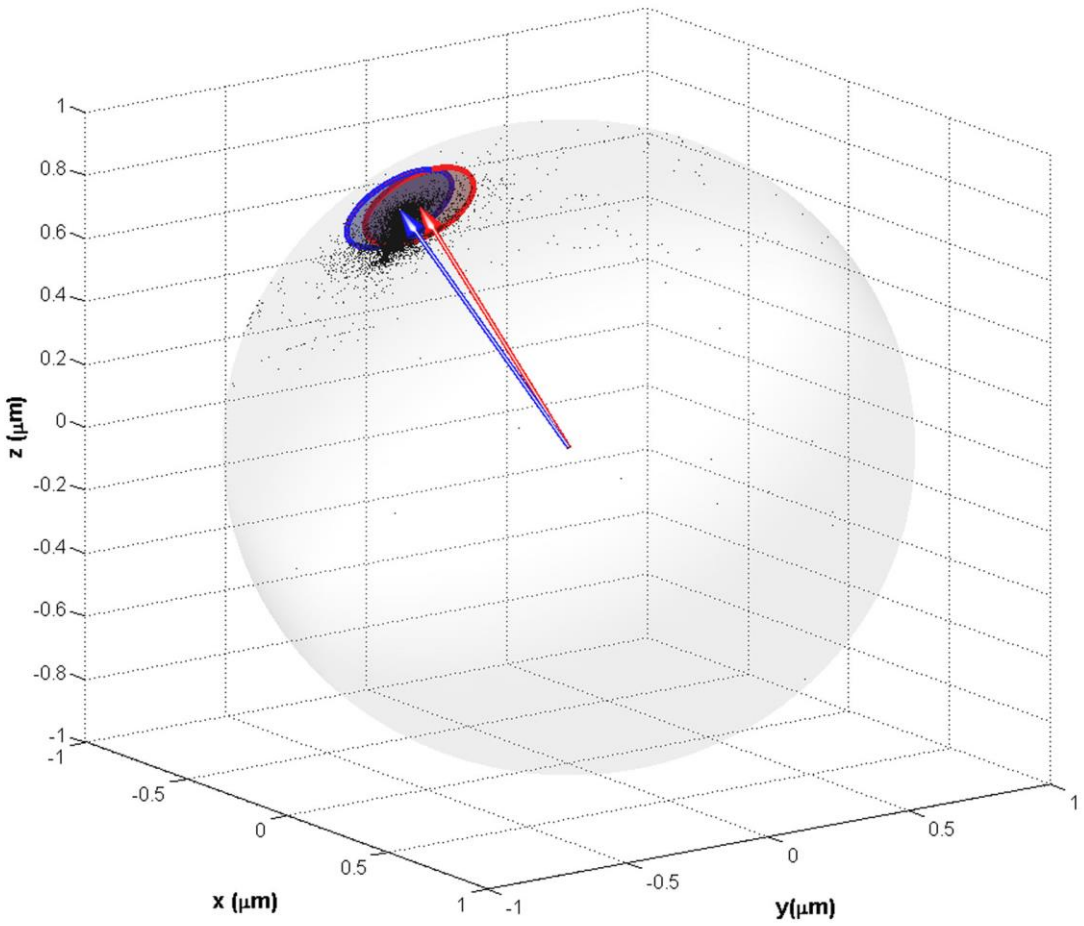


Figure 4.13 Interface element normal point clouds for the interface between the red and blue volumes shown in Figure 4.11. Red and blue arrows representing twin plane normals for the crystal volumes.

The area of the triangle elements defined as coherent, A_C, was compared to the total interface area, A, Equation 4.5:

$$\lambda_A = \frac{A_C}{A}$$

Equation 4.5

where λ_A is the fraction of interface containing coherent area per unit area. For the interface between the red and blue crystal volumes, 94% of the total interface area was coherent.

4.3 Results and Discussion

4.3.1 Correcting Grain Size Distribution through Saltykov Analysis

The 3D grain size distribution measured from the EBSD mapped serial sectioned volume is shown in Figure 4.14. Alongside is the 2D grain size distribution measured from a single EBSD serial section (section 40).

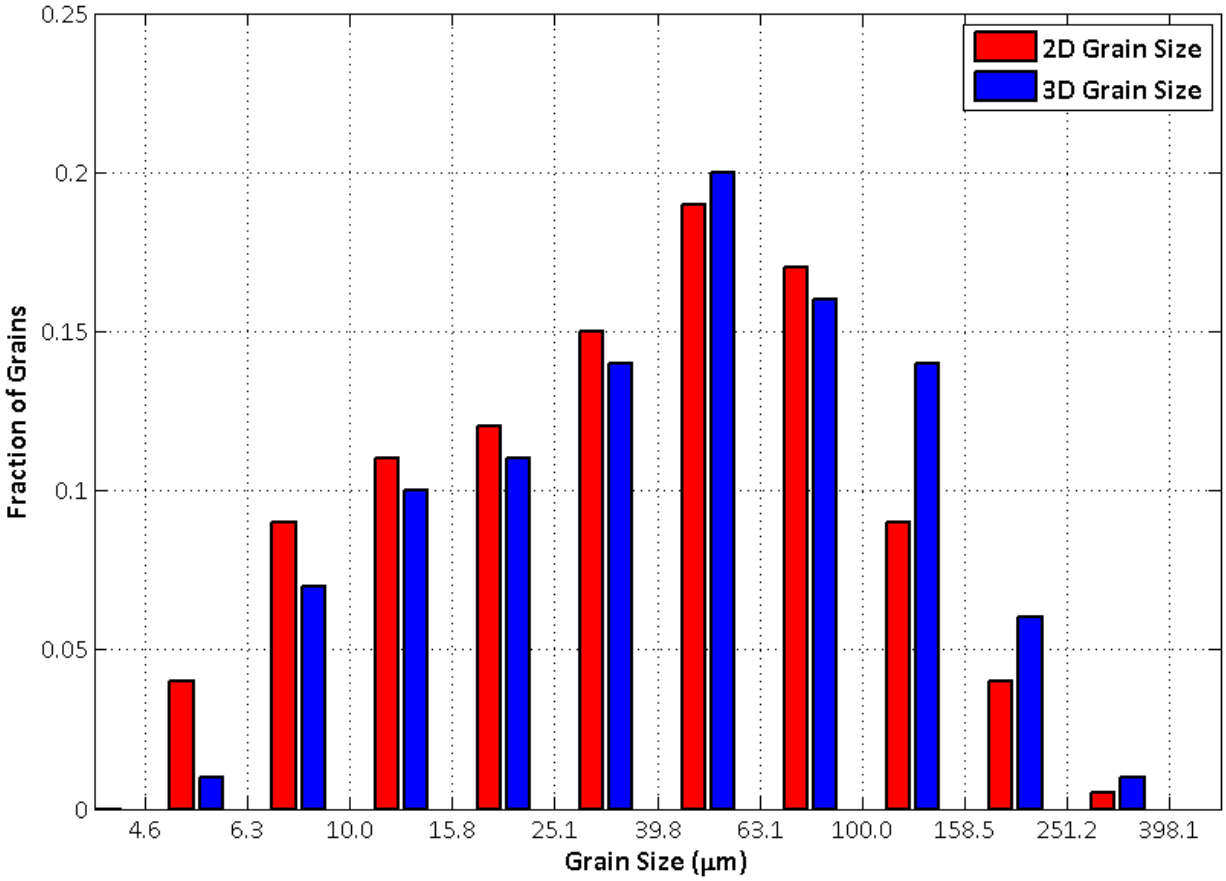


Figure 4.14 2D and 3D grain size distributions measured from the EBSD maps.

The smallest grain identified in 3D measurement process was 5.1µm which was similar to the smallest grain measured from the single EBSD section of 4.1µm. The values indicate that the noise reduction procedure, in which regions of five pixels or less are removed, is appropriate. The largest 3D measured grain was 284.2µm, approximately 4% larger than the largest grain measured from the single EBSD section, 274.5µm. While the sizes of the largest grains were comparable, Figure 4.14 shows that the 3D grain size distribution has a proportionally greater number of grains located in the upper tail. This skew towards the lower tail seen in the 2D grain size distribution results in an average grain size of 51.5µm,

approximately 25% less than the 3D measured average grain size of 64.9µm. The difference in average grain sizes is corrected by employing the Saltykov stereographic correction.

The results of the Saltykov correction are compared to the 3D measured grain size and the 2D measured grain size in Table 4.1 and Figure 4.15. Figure 4.15 highlights differences in the upper and lower tails between the distributions. Applying the Saltykov correction reduces the proportion of grains at the lower tail and increases the proportion of grains at the upper tail of the distribution. This shift in the distribution produced an average grain size of 61.0µm which is comparable (6% difference) to the measured 3D average grain size of 64.9µm.

Table 4.1 Comparison of grain size and grain size distribution statistics for all 2D measured grain size, Saltykov 2D corrected grain size, and 3D measured grain size. The *Average Grain Size* is given by Equation 3.19, the G*rain Size %Error* is given by Equation 3.20, and the *Coefficient of Variation* describing the width of the grain size distributed grain size measurements is given by Equation 3.23.

	2D Measured Grain Size	**Saltykov 2D Corrected Grain Size**	**3D Measured Grain Size**
Average Grain Size (µm)	51.5 ± 8.9%	61.0 ± 9.1%	64.9 ± 9.0%
Smallest Grain (µm)	4.1	4.1	5.1
Largest Grain (µm)	274.5	274.5	284.2
Coefficient of Variation (CV)	0.92	0.88	0.87
Number of Grains Measured	386	403	100

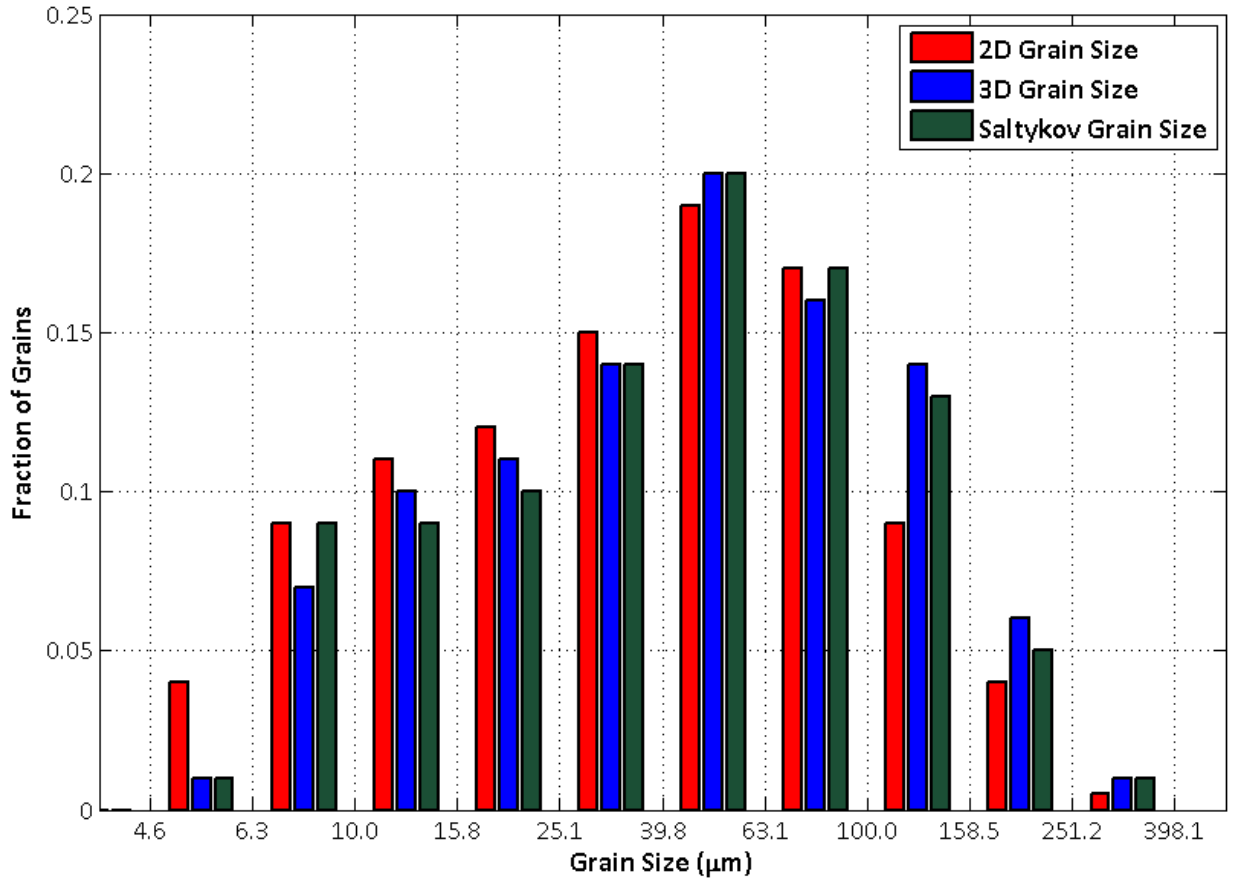


Figure 4.15 Histogram comparing 2D measured grain size data, Saltykov 2D corrected grain size, and 3D measured grain size.

4.3.2 3D Morphology

Figures 4.16 to 4.24 shows the seven 3D reconstructions produced from the optical serial sections each comprised of two or more crystal volumes. If a crystal volume contained at least one Σ3 interface, then it was called a Σ3 crystal volume. Not all the reconstructions shown are grains by this study's definition because not all the Σ3 boundaries are coherent twins. Rather, the reconstructions are an amalgamation of Σ3 crystal volumes. These amalgamations are called Σ3 clusters.

The number of interfaces belonging to the $\Sigma3^n$ (n = 1, 2, 3, and 4) rotation group are summarised in Table 4.2 for each of the seven reconstructions. Three of the clusters contained Σ3 boundaries only, while the other four also contain higher-order $\Sigma3^n$ variants (Σ9, Σ27, and Σ81 boundaries).

Table 4.2 $\Sigma3^n$ boundary types for each volume cluster.

Σ3 Cluster	**Number of Σ3 Volumes**	**Number of $\Sigma3^n$ Interfaces**				**Total Number of Interfaces**
		Σ3	*Σ9*	*Σ27*	*Σ81*	
1	4	3				**3**
2	10	9	1			**10**
3	5	4	3			**7**
4	3	2				**2**
5	3	2				**2**
6	7	7	2			**9**
7	38	48	35	17	5	**105**
Total	**70**	**75**	**41**	**17**	**5**	**138**

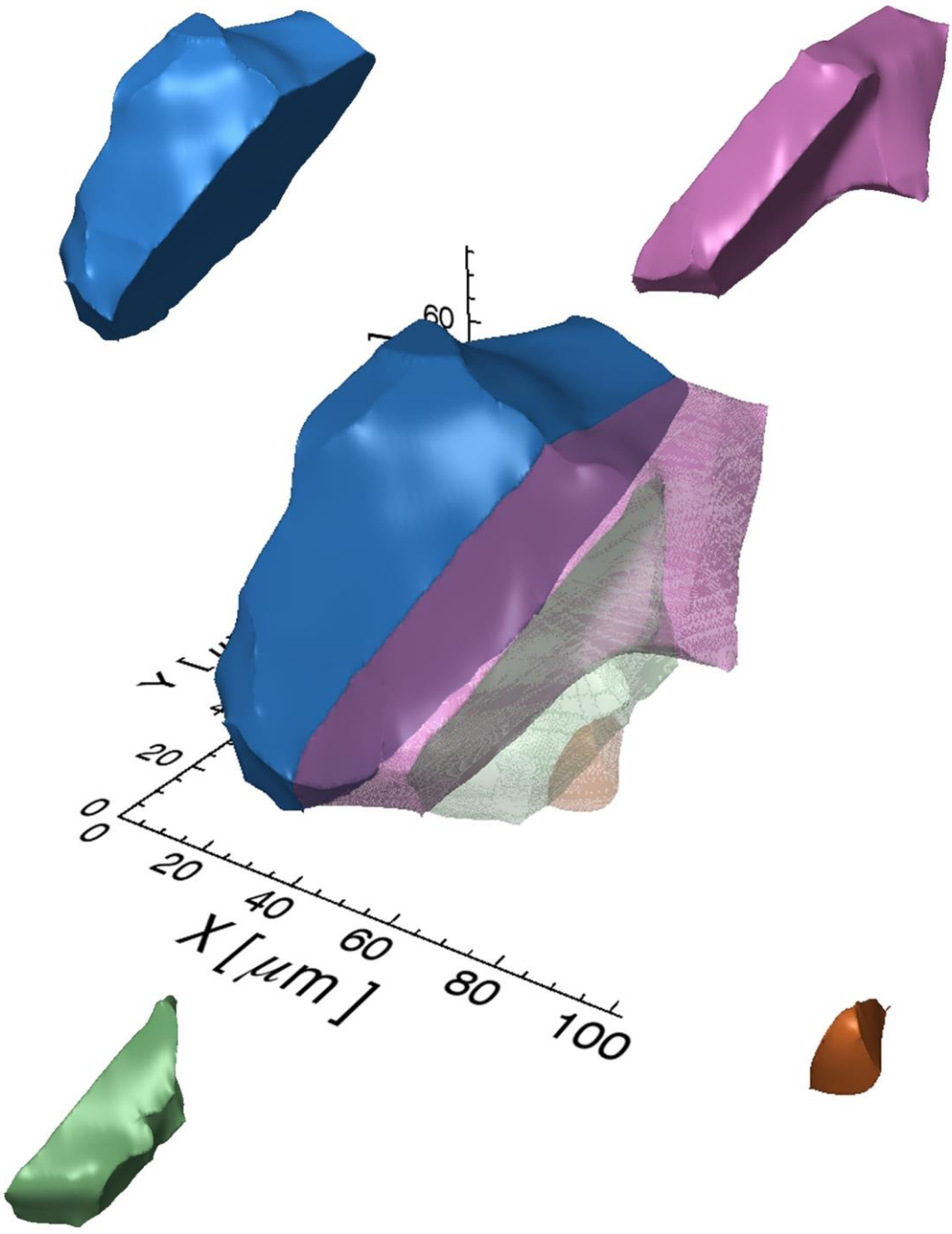


Figure 4.16 3D reconstruction of Σ3 Cluster 1 comprised of four Σ3 crystal volumes.

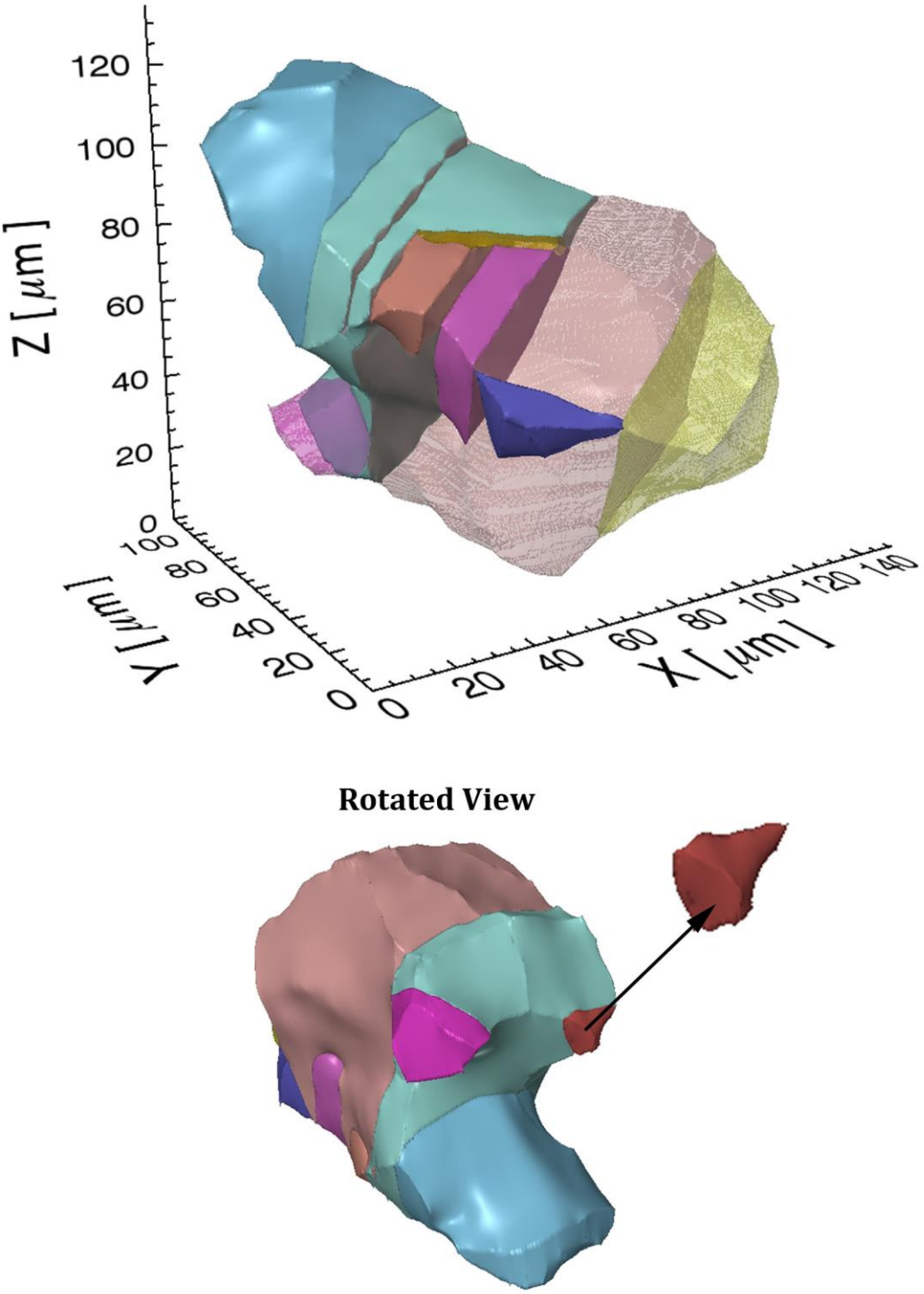


Figure 4.17 3D reconstruction of Σ3 Cluster 2 comprised of ten Σ3 crystal volumes.

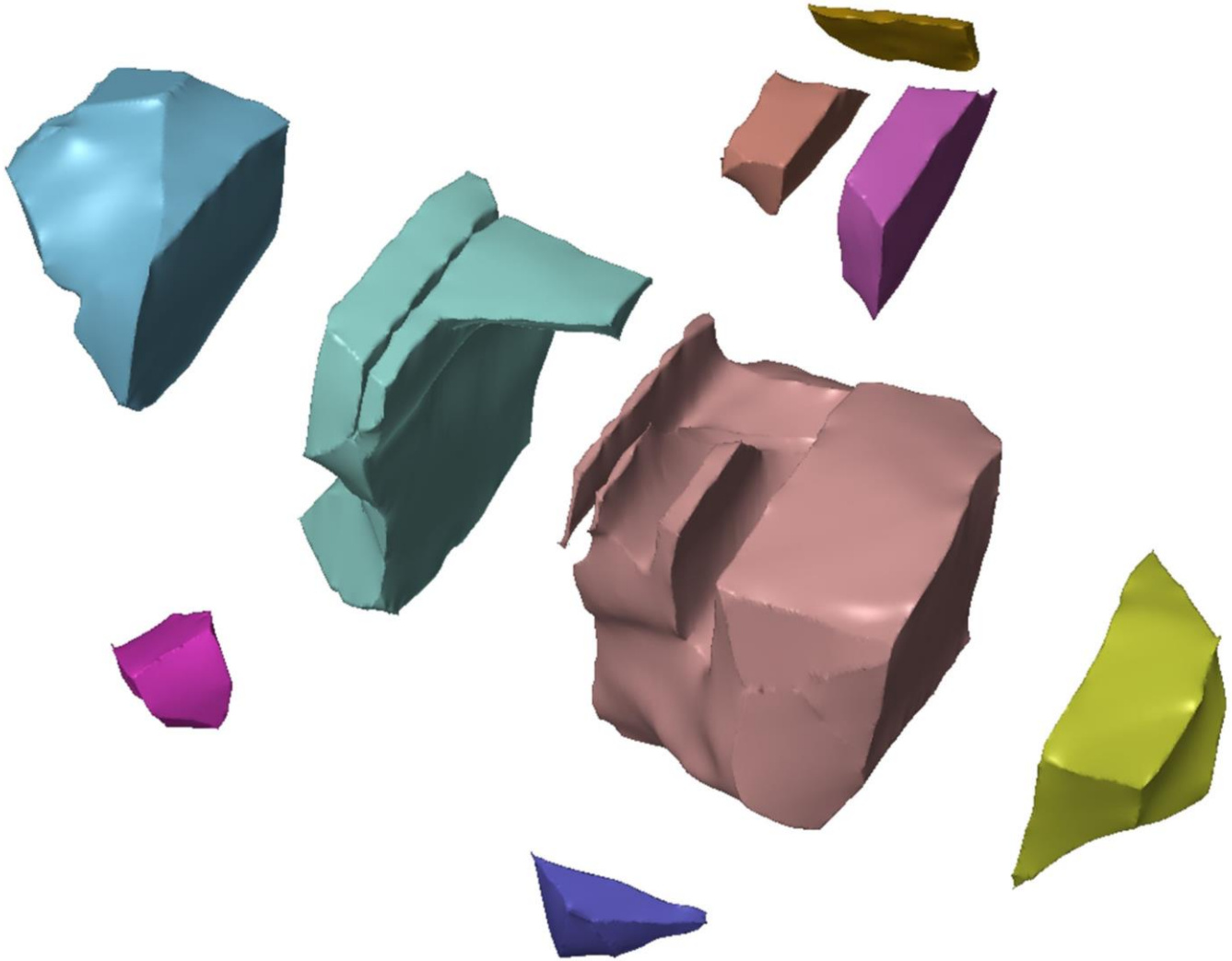


Figure 4.18 An exploded view of Σ3 Cluster 2.

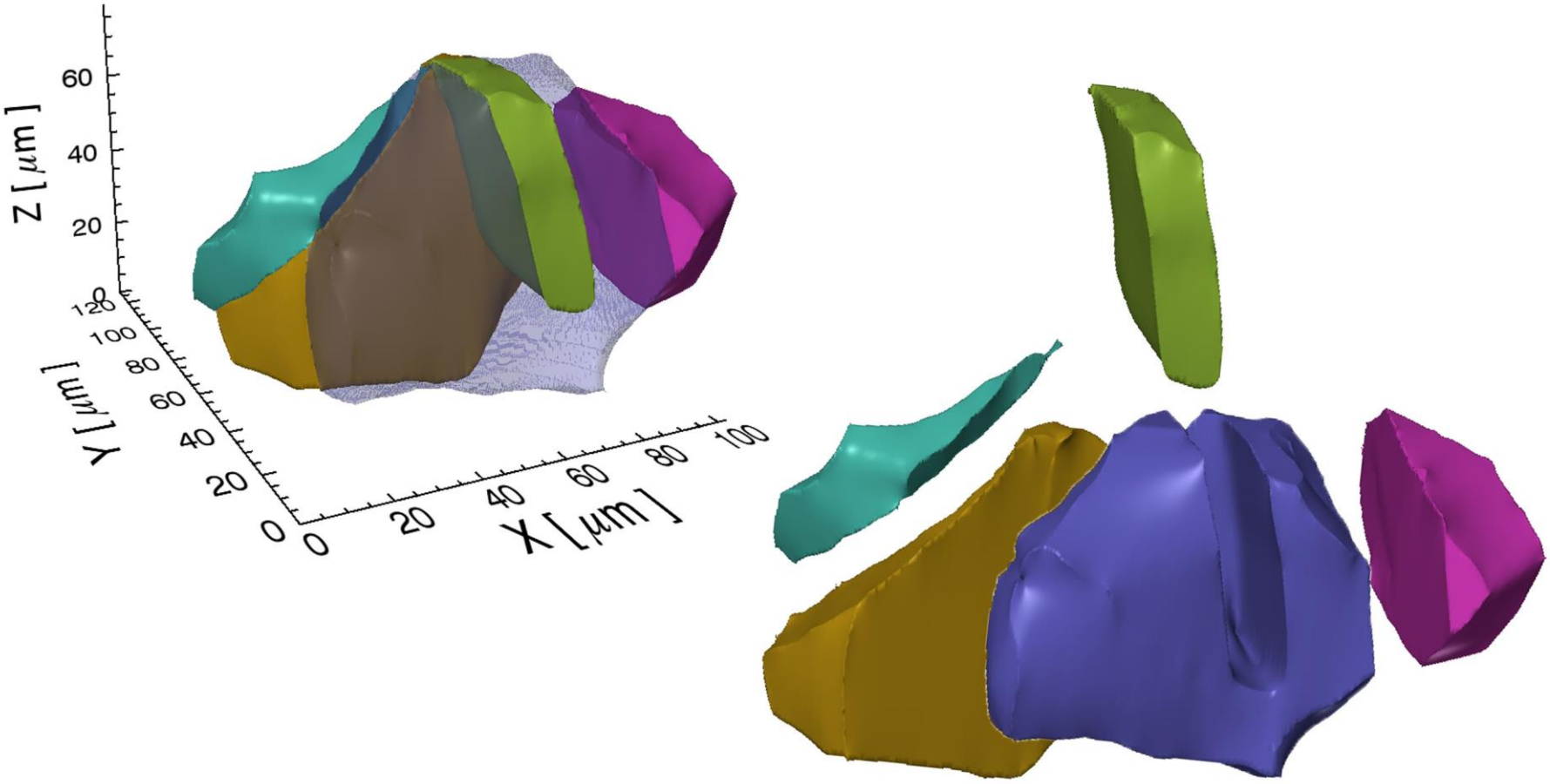


Figure 4.19 3D reconstruction of Σ3 Cluster 3 comprised of five Σ3 crystal volumes.

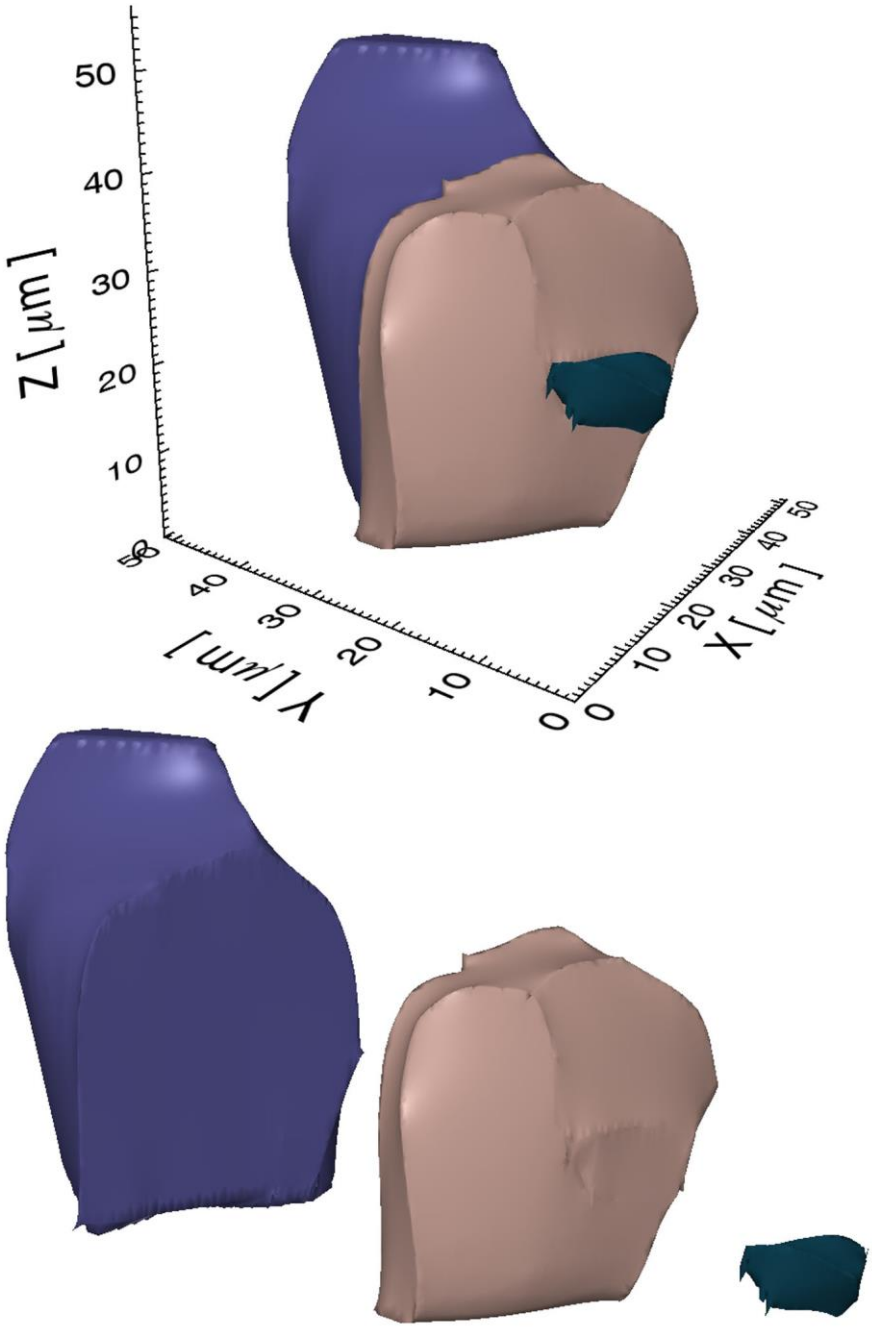


Figure 4.20 3D reconstruction of Σ3 Cluster 4 comprised of three Σ3 crystal volumes.

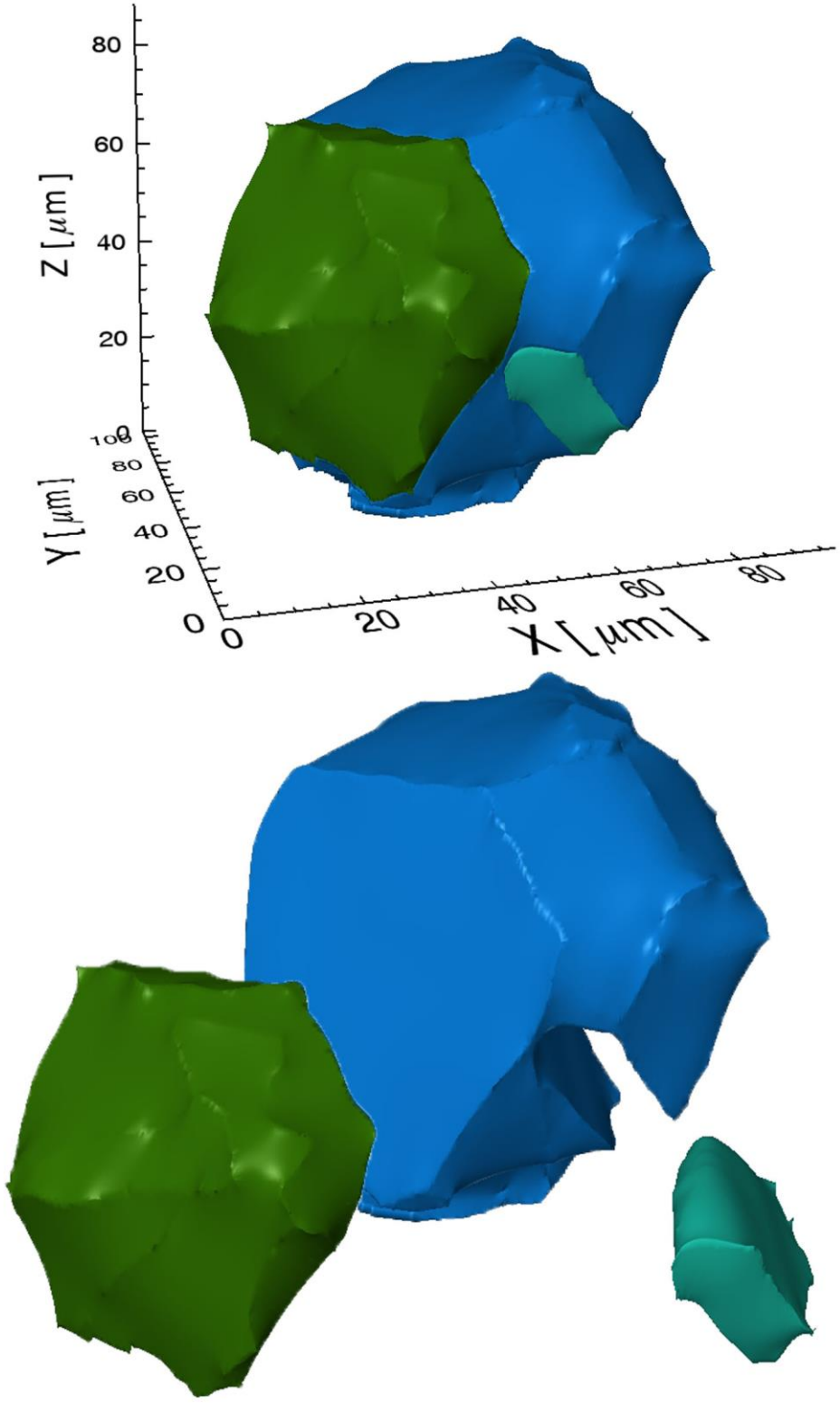


Figure 4.21 3D reconstruction of Σ3 Cluster 5 comprised of three Σ3 crystal volumes.

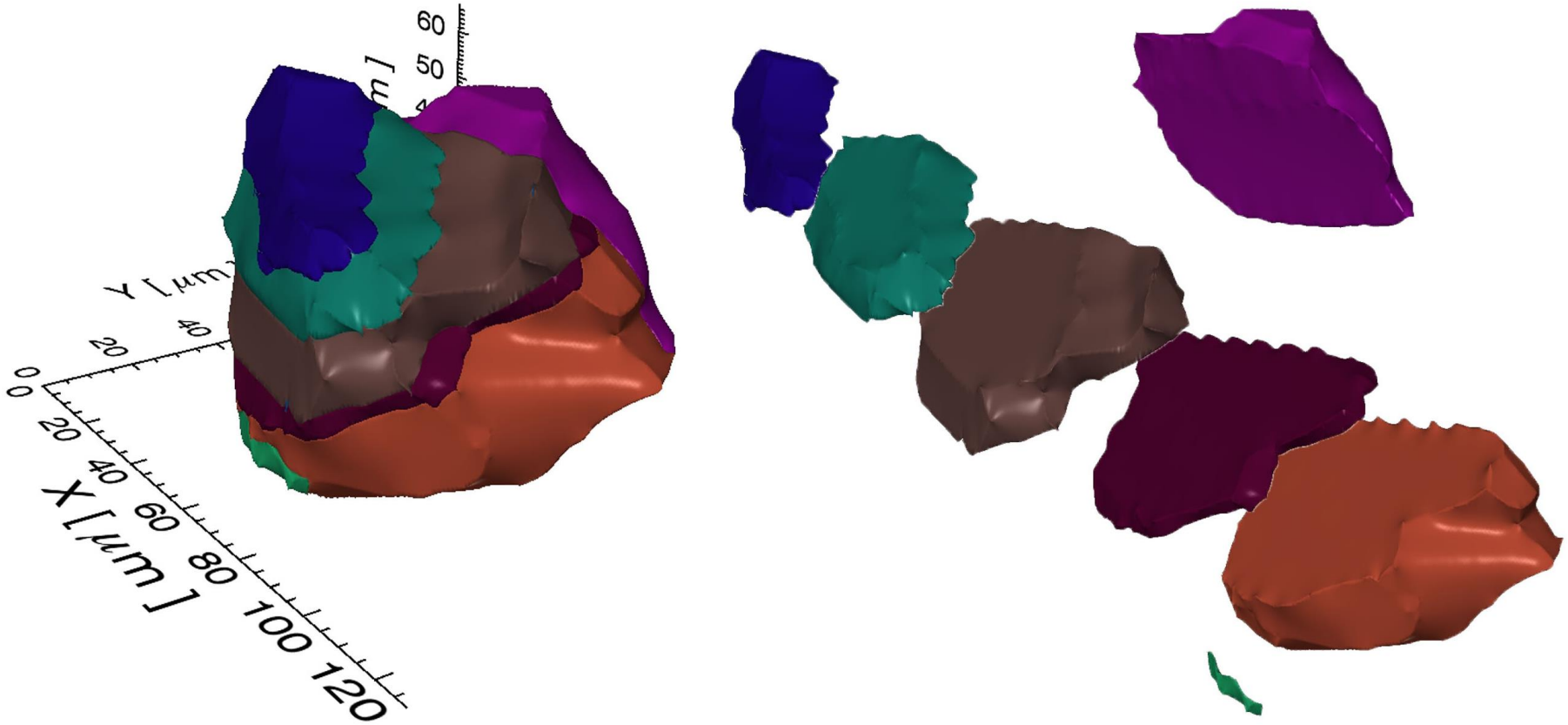


Figure 4.22 3D reconstruction of Σ3 Cluster 6 comprised of seven Σ3 crystal volumes.

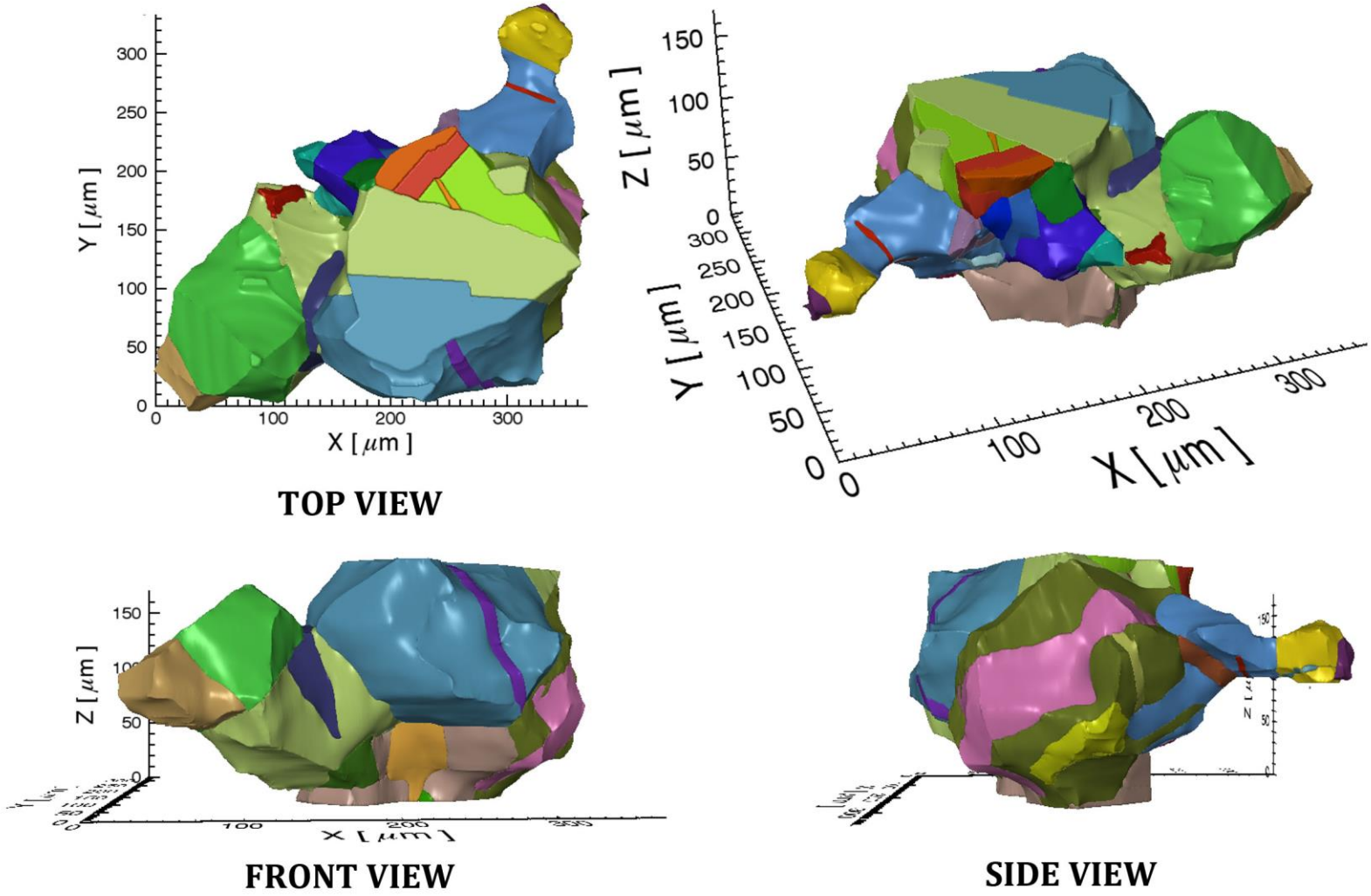
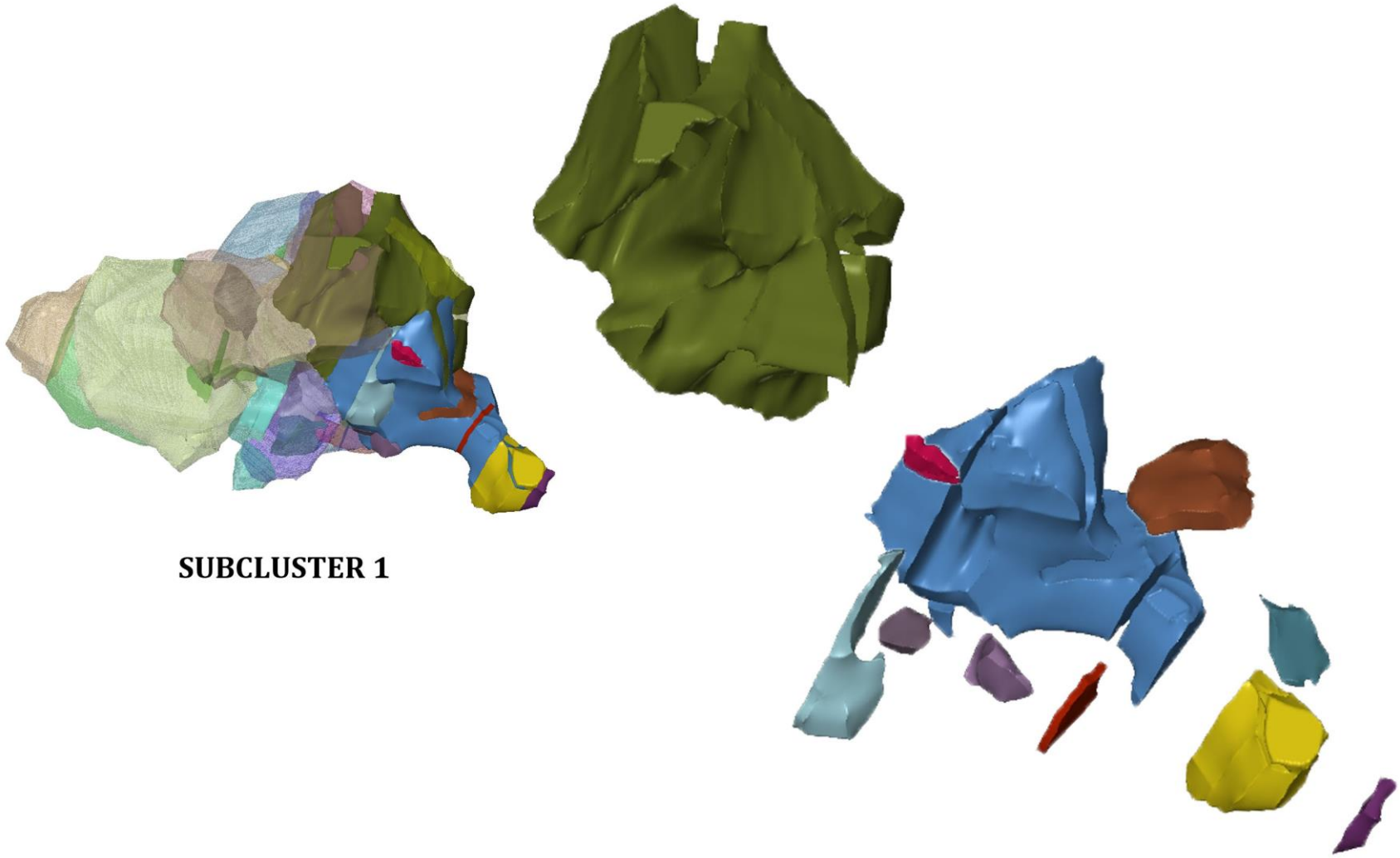


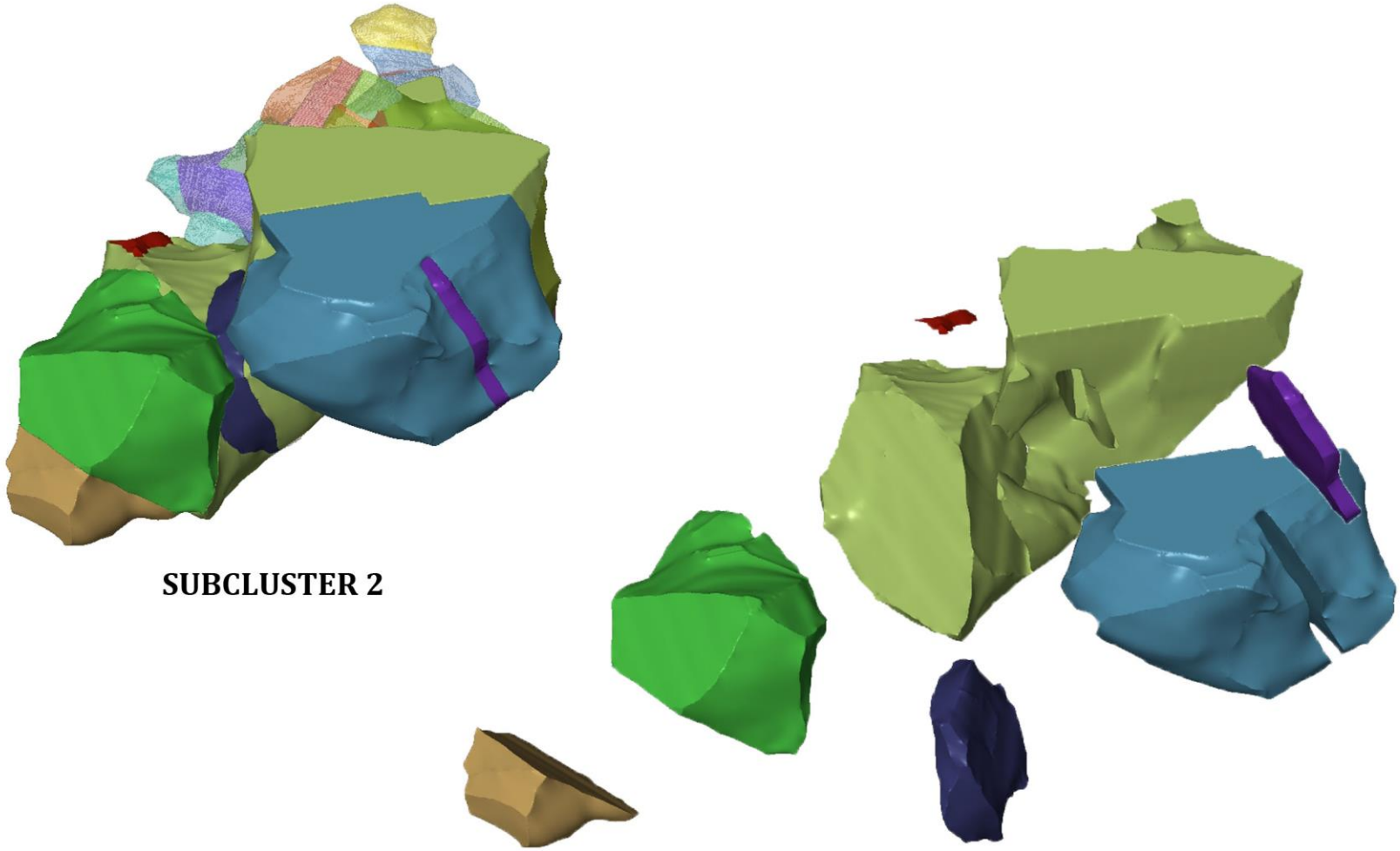
Figure 4.23 3D reconstruction of Σ3 Cluster 7 comprised of thirty-eight Σ3 crystal volumes.

(a)



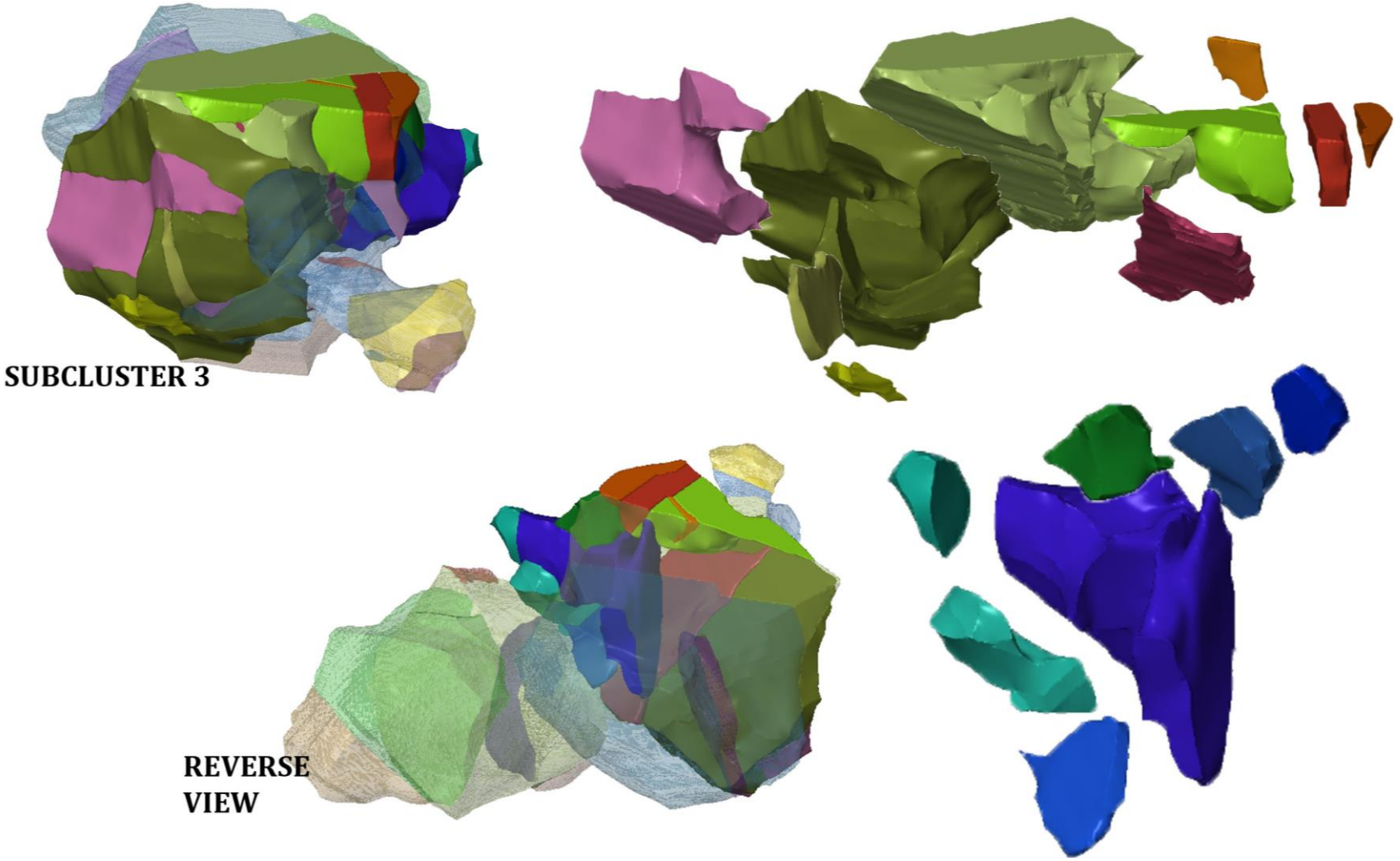
SUBCLUSTER 1

(b)



SUBCLUSTER 2

(c)



(d)

SUBCLUSTER 4

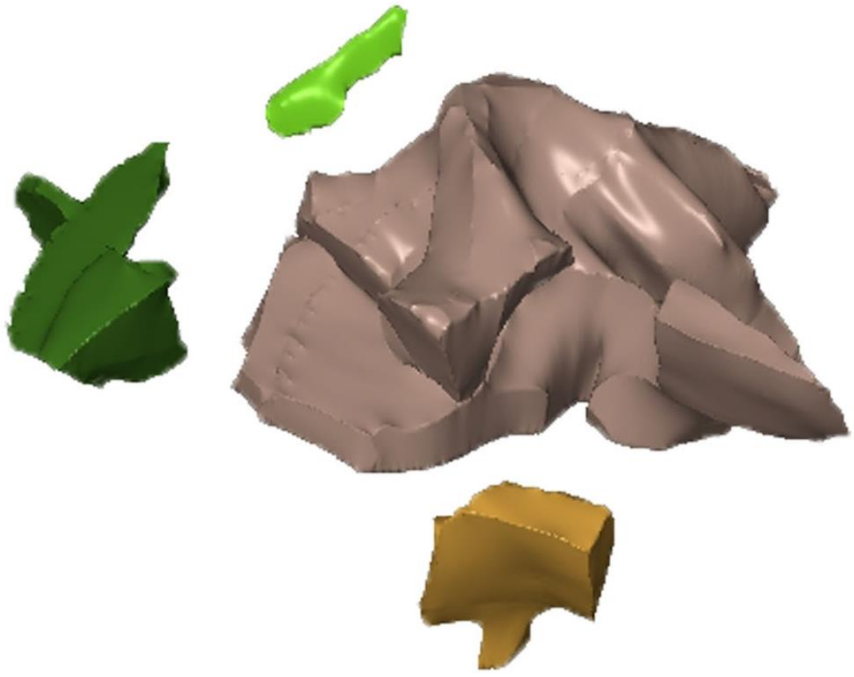


Figure 4.24 Exploded views from four (a-d) Σ3 crystal volume subclusters from 3D reconstruction of Σ3 Cluster 7. Subclusters are used to assist in the visualization of the 3D volumes.

The 70 reconstructed Σ3 crystal volumes produce a range of morphologies that can be assigned an initial categorization of *simple* or *complex*. The 2D and 3D morphologies of simple volumes were previously examined by Bystrzycki et al [73]. Bystrzycki provided descriptions in agreement with the morphologies observed in the current investigation. In both Bystrzycki's and the current analysis, *simple* volumes were further categorized as lamellar or edge structures. Lamellar structures are characterized by the appearance of parallel Σ3 boundary planes, Figure 4.25(a), while an edge structure contains a single Σ3 interface, Figure 4.25(b).

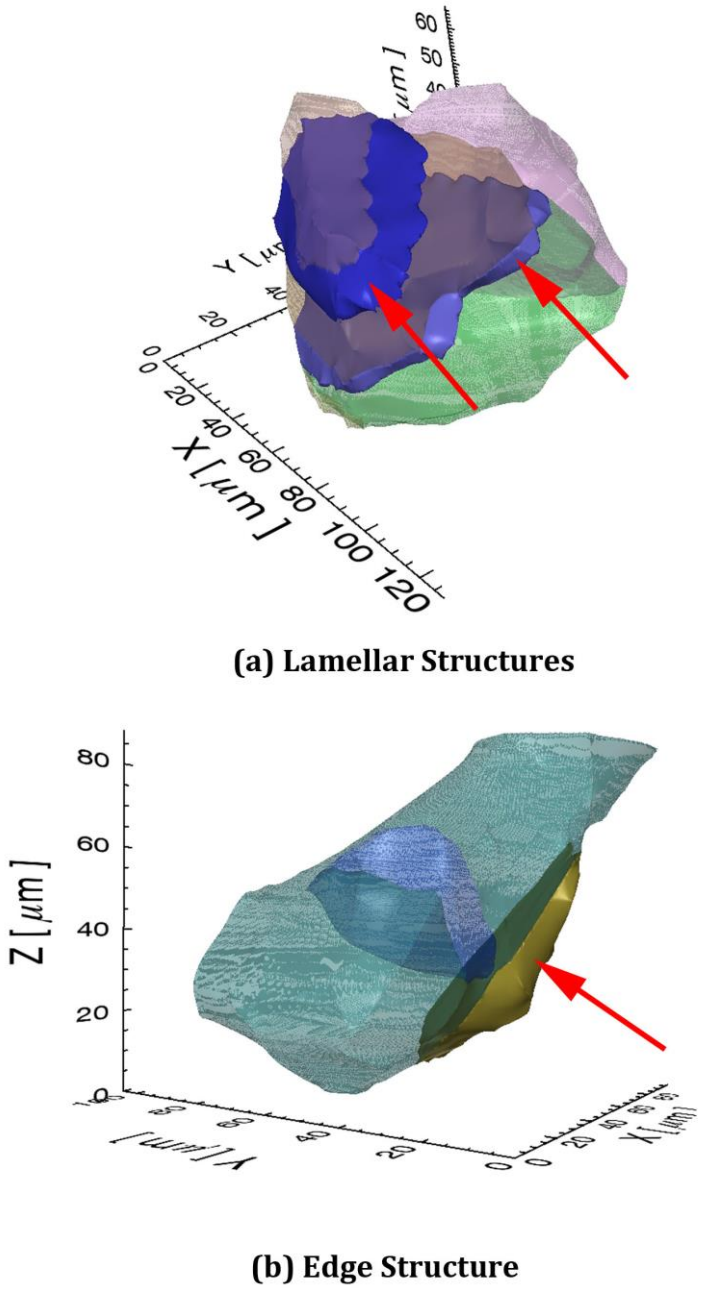


Figure 4.25 Simple volume structures: (a) lamellar and (b) edge.

The sectioning plane location can alter the perception of morphology. Figure 4.26 shows the sectioning of two simple volumes, Volume 1 and Volume 2. The 3D morphologies are categorised as lamellar and edge structures for volumes 1 and 2 respectively. Figure 4.26(a) shows the volumes sectioned in such a way that the resulting 2D morphologies would be described as an incomplete parallel Σ3 and a one sided Σ3. Figure 4.26(b) shows that when the section is made at a different location, twin volume 1 is now characterized as a one sided Σ3 in 2D. The observation suggests that, while the terms lamellar and edge can be used to describe simple 3D volumes, the position of the sectioning plane will alter the 2D perceptions.

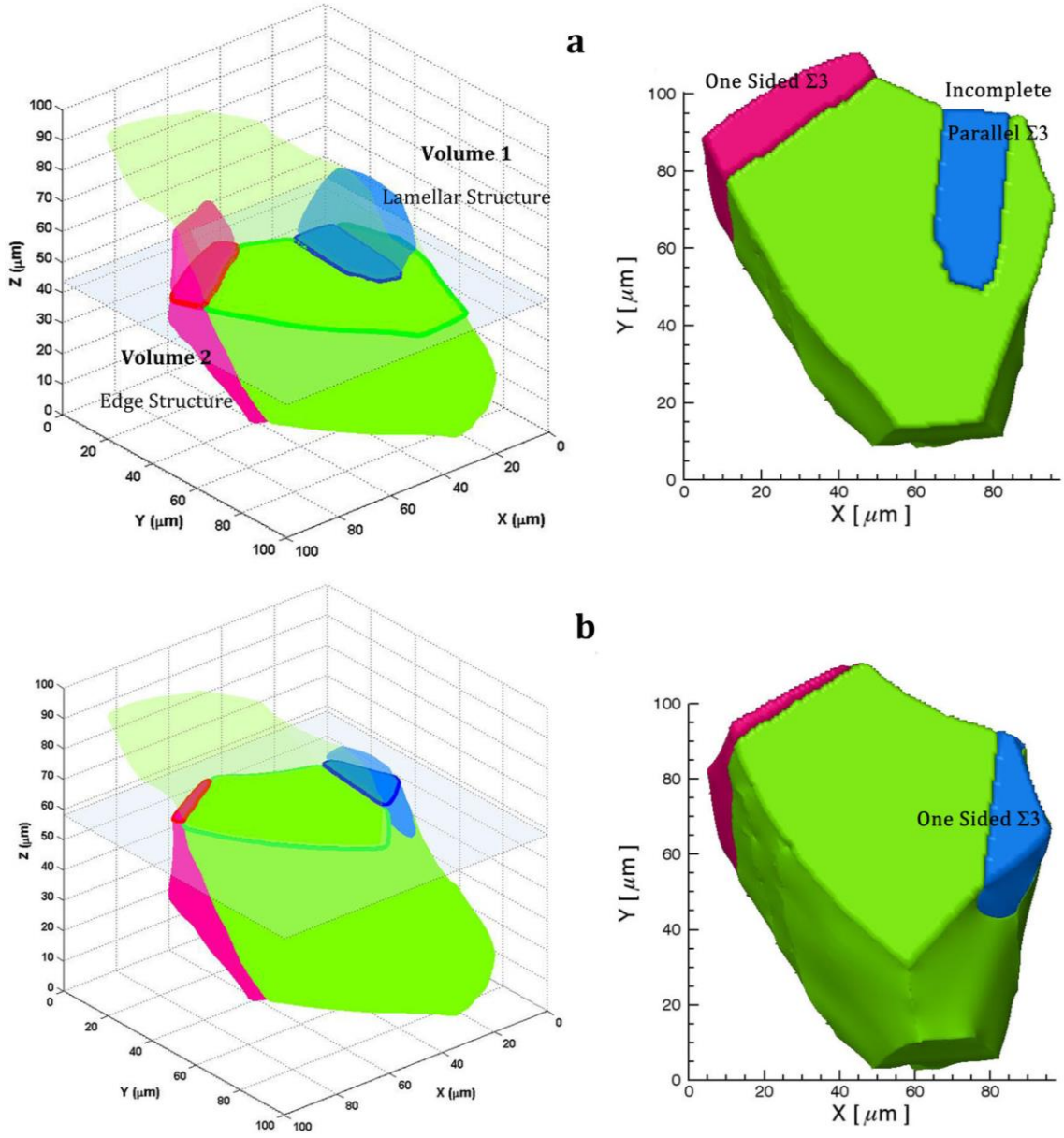


Figure 4.26 Simple twin volumes (Volume 1 and Volume 2) from Cluster 5 sectioned in two locations (a and b) resulting in a change in the representation of 2D morphology.

When multiple Σ3 boundaries impinge on each other, the resulting volumes can become complex. A *complex* volume has a morphology that can no longer be characterised with simple terms such as edge or lamellar. These complex Σ3 volumes were discussed by Reed et al [1] and were described as possessing highly re-entrant shapes that can intersect a sectioning plane many times, giving a false impression of multiple, separate boundaries. Figure 4.27 is a complex volume from cluster 7 and shares $\Sigma 3^n$ interfaces with 12 neighbouring volumes.

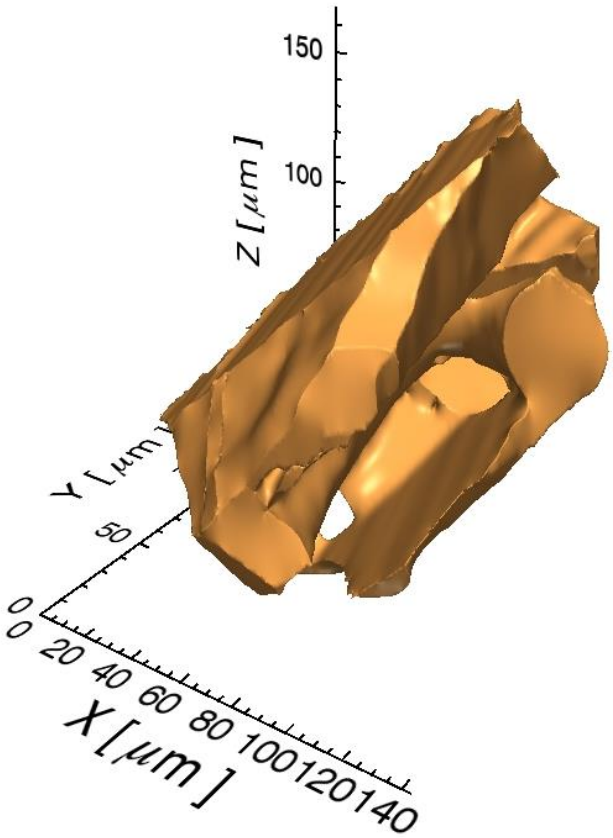
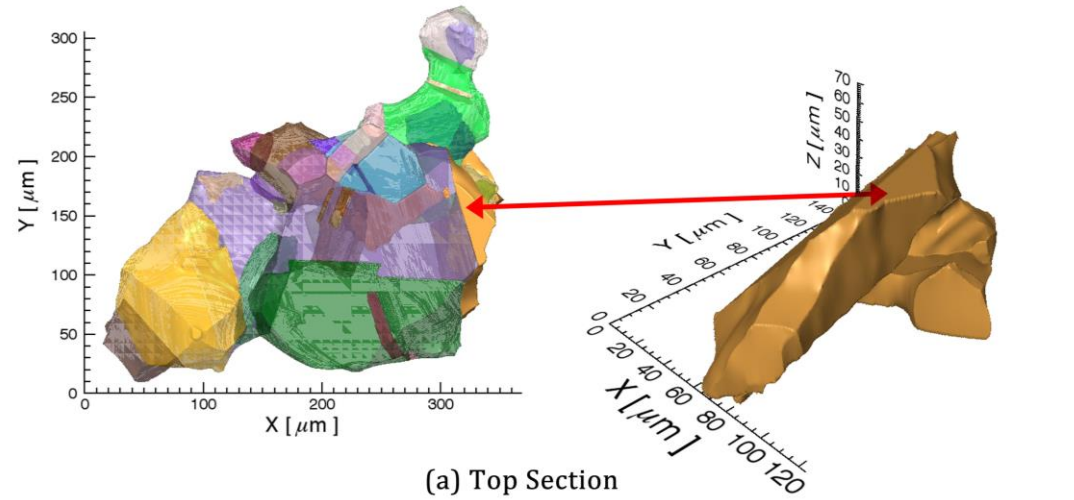


Figure 4.27 A complex twin volume.

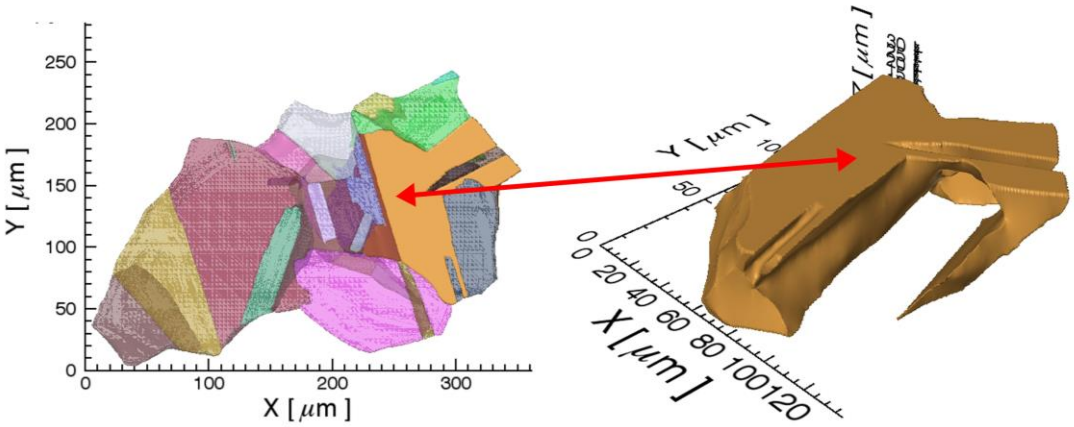
When sectioned, these complex volumes may present a 2D morphology similar to those mentioned above. However, these simple 2D representations are far removed from the morphologies observed in 3D. Even when the 2D morphology representing the volume does indicate at least some complexity, for example Figure 4.28(b), it is often the case that many of the re-entrant and protruding morphologies are not represented.

Figure 4.28 shows a complex volume with 2D morphologies produced by sectioning at three locations. Figure 4.28(a) shows the volume sectioned at the top with the resulting 2D morphology that could be described as a one sided Σ3. Figure 4.28(b) shows the volume sectioned near its centre where the resulting 2D morphology is more representative of the volume's complexity, with many of the re-

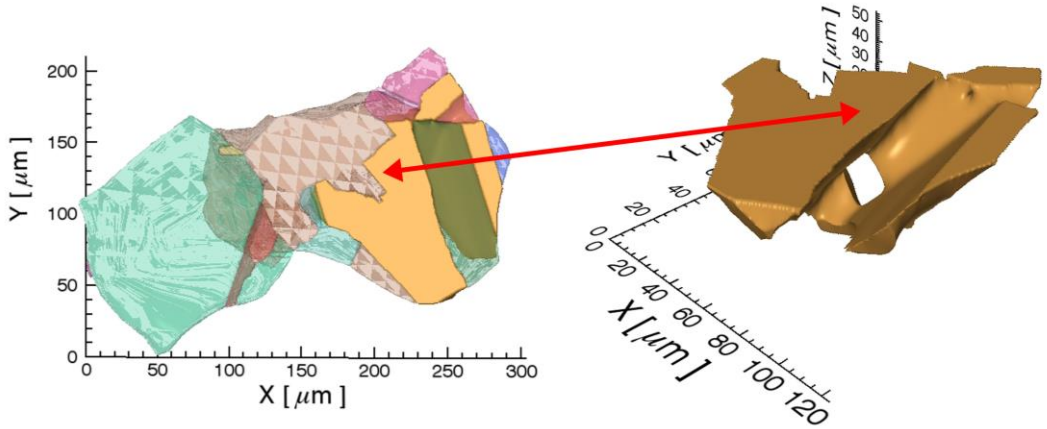
entrant geometries revealed. Figure 4.28(c) shows the twin volume sectioned towards its base where the resulting 2D morphology is separated into three areas, giving the impression of the presence of three distinct twin volumes.



(a) Top Section



(b) Middle Section



(c) Bottom Section

Figure 4.28 Complex volume from Figure 4.27 sectioned in three locations (a-c) showing the variety of 2D morphologies that are possible.

An island twin was one of the four 2D descriptors used to describe twin morphology [64]. The term island twin suggests a volume exists whereby there is no connectivity with the HGB network. While twin boundaries can be represented in 2D with no connectivity, it was found at all times that further sectioning revealed that the twin volume was connected to the HGB network.

Observing the variety of complex morphologies that may be present within a microstructure is vital for understanding the intricate, and sometimes confusing, topology of grain boundary networks that appear on 2D sections.

4.3.3 Identifying Coherent Σ3 Boundaries

The interface indices of the triangular mesh elements were measured for each of the 75 reconstructed Σ3 boundaries and were used to access coherency. The fraction of boundaries containing coherent length per unit length in 2D, λ_l, and the fraction of interface containing coherent area per unit area in 3D, λ_A, is presented in Figure 4.29. λ_l was measured for Σ3 boundaries in all EBSD maps from the sectioned volume using trace analysis.

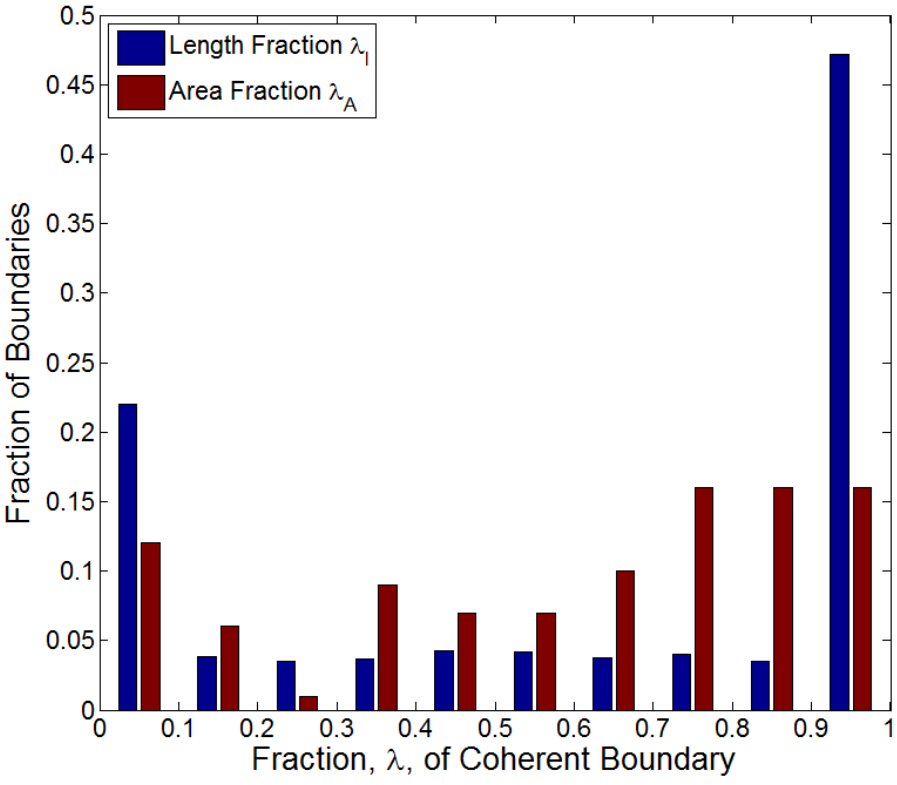


Figure 4.29 The fraction of Σ3 boundaries containing coherent length per unit length in 2D, λ_l, and the fraction of interfaces containing coherent area per unit area in 3D, λ_A.

For the 3D and 2D measurements, local maxima are present at the fully non-coherent (left) and fully coherent (right) boundaries, with fully coherent dominating. For the 3D reconstructed interfaces, the observed fractions for the fully non-coherent and fully coherent boundaries were significantly less than that measured in two-dimensions via trace analysis; possible reasons for this discrepancy follow.

Imperfect section alignment and interface surface meshing can often lead to the production of artifacts. The triangular mesh elements at these artifacts may be measured as non-coherent, when in fact the interface is coherent. Conversely, an interface that is fully non-coherent may have several coherent Σ3 elements.

The boundary reconstruction process may reconstruct a boundary with a single line segment even if the boundary is not perfectly straight, as long as the line segment is within the specified tolerance to the mapped EBSD boundary. This characteristic of the boundary reconstruction algorithm produces boundaries that could only be defined as 100% coherent or 100% non-coherent, resulting in a skew towards the tails of the distribution.

While approximately 65% of the total Σ3 boundary population was determined to be either fully coherent or fully non-coherent via trace analysis, 35% of boundaries had $0.1 < \lambda_l \leq 0.9$. This result posed the question, what proportion of a boundary trace should be identified as coherent in order for a boundary to be considered a coherent twin? An answer to this question was required for the identification of coherent twins to define a grain for grain size measurement.

A sensitivity analysis was performed on the effect of coherency on average grain size. For this analysis, the average grain size for the EBSD serial sections was recalculated for coherency values of $0.1 < \lambda_l \leq 0.9$. The results, Figure 4.30, showed that the effect on grain size was less than 10% across the range, which is acceptable for grain size measurement [3]. For the current study, a threshold of $\lambda_l = 0.5$ was employed to define a coherent twin boundary, a value also considered appropriate in a separate study [145].

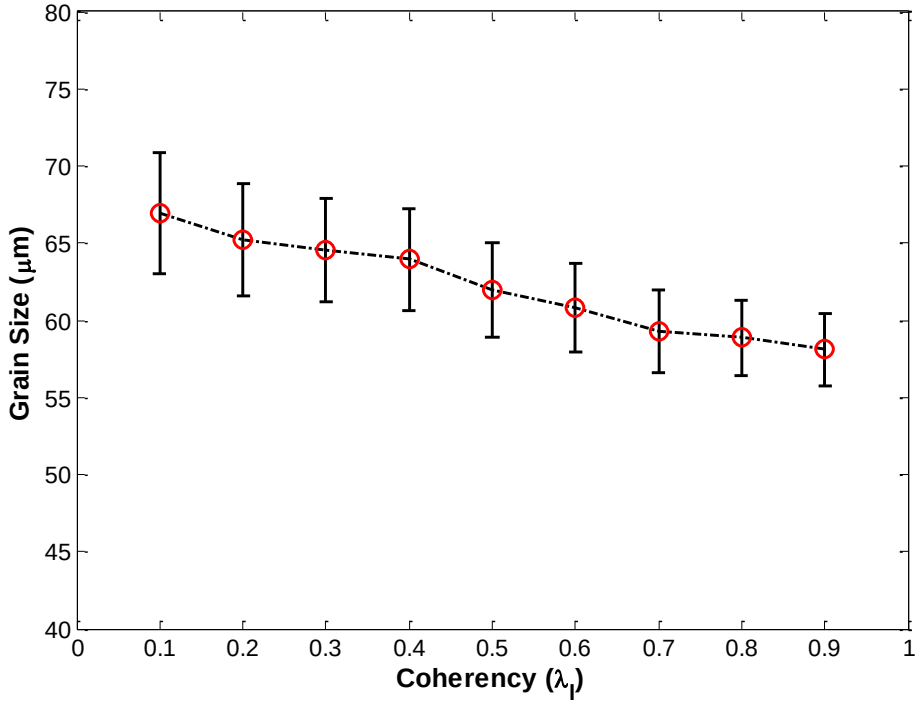


Figure 4.30 Change in average grain size for coherency threshold ranging from 0.1 to 0.9.

4.3.4 Boundary Faceting

Σ3 boundaries often display a stepped appearance in optical images. As shown in Figure 4.31, these Σ3 boundaries are composed of multiple planar interfaces, broad ledge interfaces and smaller risers (1-2μm).

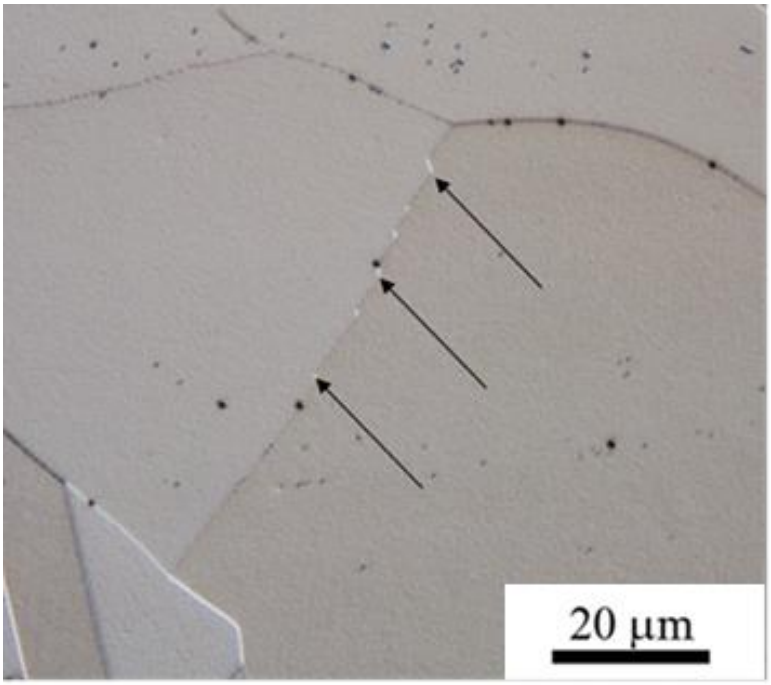


Figure 4.31 Optical image of a faceted Σ3 boundary with some risers indicated with arrows.

The stepped appearance of boundaries is now discussed in relation to our understanding of the formation of Σ3 boundaries during recrystallization:

1. The formation of Σ3 boundaries occur at migrating grain boundaries.
2. The formation of a Σ3 boundary reduces total grain boundary energy.
3. The formation of a Σ3 boundary increases the growth rate of the recrystallizing nucleus.

Σ3 boundaries form with geometries favourable to the minimization of interface energy, and in accordance with Young's Law [146], Equation 4.6:

$$\frac{\gamma_1}{\sin \varepsilon_1} = \frac{\gamma_2}{\sin \varepsilon_2} = \frac{\gamma_3}{\sin \varepsilon_3}$$

Equation 4.6

where γ_j and ε_j are the grain boundary energies and dihedral angles respectively, Figure 4.32. For boundaries of similar energies it is expected that the three dihedral angles will be approximately equal

(≈ 120°). This is typically observed at the intersection of three RHGB where it can be assumed that the boundary energy is independent to the interface inclination.

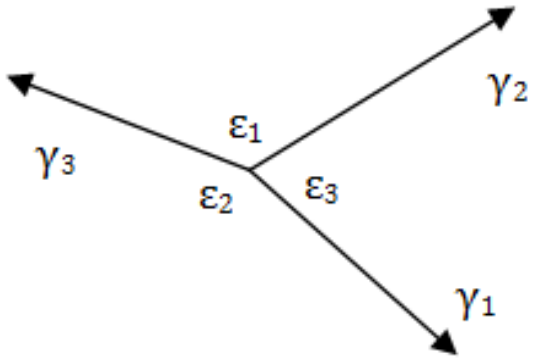


Figure 4.32 Schematic illustration of a planar section with three boundaries of energies γ_1, γ_2, and γ_3, and dihedral angles ε_1, ε_2, and ε_3.

Figure 4.33 is a schematic representation showing the formation of a Σ3 boundary parallel to the recrystallization front. Figure 4.33(a) shows the migration, V_1, of a RHGB through the deformed microstructure during recrystallization. Figure 4.33(b) shows the RHGB encountering a deformed region of similar orientation to the recrystallizing grain, resulting in a boundary misorientation no longer suitable for continued migration, $V_2 < V_1$. Figure 4.33(c) shows the decomposition of the RHGB into a Σ3 boundary and a boundary providing favourable misorientation (new boundary identified as *'1-2'*) for recrystallization, $V_3 > V_2$.

Equation 4.7 represents the energy balance for the formation of the Σ3 boundary without a resulting increase in total boundary energy:

$$\gamma_{\Sigma 3} A_{\Sigma 3} + \gamma_{1-2} A_{1-2} < \gamma_{RHGB} A_{RHGB}$$

Equation 4.7

where $\gamma_{\Sigma 3}$, γ_{1-2}, and γ_{RHGB} are the energy per unit of area of the boundaries, and $A_{\Sigma 3,}$ A_{1-2}, and A_{RHGB} are the interface areas of the boundaries. Solving Equation 4.7 immediately after the formation of the Σ3, i.e. assuming that $A_{RHGB} \approx A_{1-2}$:

$$\gamma_{\Sigma 3} \frac{A_{\Sigma 3}}{A_{RHGB}} + \gamma_{1-2} < \gamma_{RHGB}$$

Equation 4.8

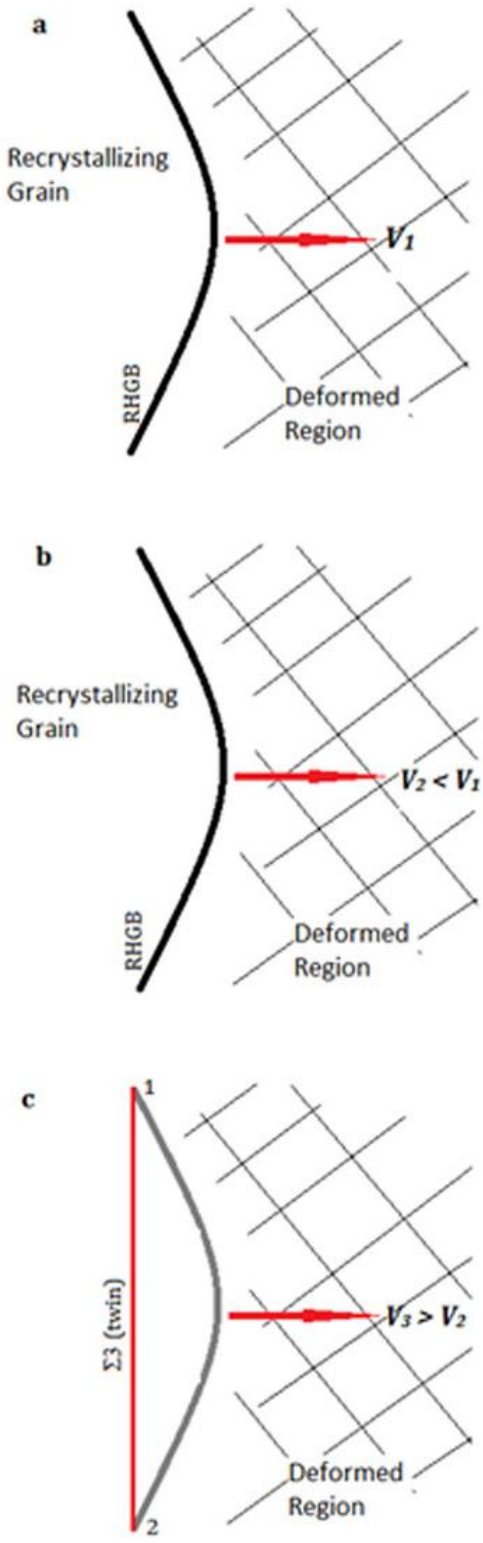


Figure 4.33 Formation of a Σ3 parallel to recrystallization front.

(a) Recrystallizing grain with RHGB migrating at velocity V_1.

(b) Recrystallizing grain with RHGB migrating at velocity $V_2 < V_1$.

(c) Σ3 formed in the recrystallizing grain resulting in an increase in boundary mobility, $V_3 > V_2$.

Kumar et al [109] showed the migration of a Σ51a boundary through a deformed region in INCONEL 600 decomposing into a Σ3 and Σ17a. While Σ17a boundaries (and other CSL boundaries, i.e. Σ7 and Σ13) are known to promote boundary mobility in fcc materials [40], they do not always present the deep energy cusps typically observed with other CSL boundaries (i.e. Σ3). Because there may be only a minimal energy difference between γ_{1-2} and γ_{RHGB}, it becomes necessary that $\gamma_{\Sigma 3}A_{\Sigma 3}$ is minimized to satisfy Equation 4.8. Assuming the Σ3 is coherent ($\Sigma 3_C$), then $\gamma_{\Sigma 3}$ (= $\gamma_{\Sigma 3C}$) is minimized and Equation 4.8 is satisfied.

Next, the geometry of the triple point, Figure 4.34, is assessed to ensure Equation 4.9 (Young's equation) is satisfied:

$$\frac{\gamma_{\Sigma 3}}{\sin \varepsilon_{\Sigma 3}} = \frac{\gamma_{1-2}}{\sin \varepsilon_{1-2}} = \frac{\gamma_{RHGB}}{\sin \varepsilon_{RHGB}}$$

Equation 4.9

Assuming $\gamma_{\Sigma 3} = \gamma_{\Sigma 3C} = 19\text{mJm}^{-2}$ and $\gamma_{RHGB} = 835\text{mJm}^{-2} \approx \gamma_{1-2}$, then the dihedral angles will be $\varepsilon_{\Sigma 3} = \varepsilon_{\Sigma 3C} \approx 179°$ and $\varepsilon_{RHGB} \approx \varepsilon_{1-2} \approx 90.5°$. Clearly, the triple point geometry illustrated in Figure 4.34 would not satisfy Young's Laws.

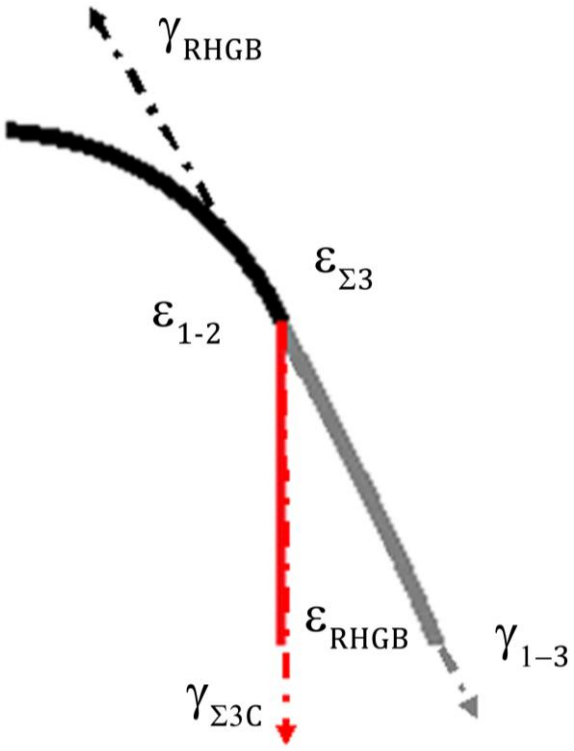


Figure 4.34 Geometry of triple point formed by coherent Σ3.

However, two arrangements may exist that satisfy both Equations 4.8 and 4.9.

1. The Σ3 dissociates into coherent and non-coherent facets, Figure 4.35.
2. The Σ3 forms a vicinal-to-coherent interface, Figure 4.36.

Figure 4.35 illustrates the dissociation of the Σ3 boundary into coherent ($\Sigma3_C$) ledges and non-coherent ($\Sigma3_N$) risers, producing a new triple point between the three boundaries. The triple point geometry is more likely to satisfy Young's condition because $\varepsilon_{1-2} \approx \varepsilon_{RHGB}$. The formation of risers increases the interface area ($A_{\Sigma3}$), with the additional non-coherent interfacial area, $A_{\Sigma3N}$, which has energy of an order of magnitude higher than a coherent Σ3. Because the increase in area associated with higher energy inclinations is unfavourable, only a limited amount of risers would be permitted before Equation 4.10 is no longer valid.

$$(\gamma_{\Sigma3C}A_{\Sigma3C} + \gamma_{\Sigma3N}A_{\Sigma3N}) < A_{RHGB}(\gamma_{RHGB} - \gamma_{1-2})$$

Equation 4.10

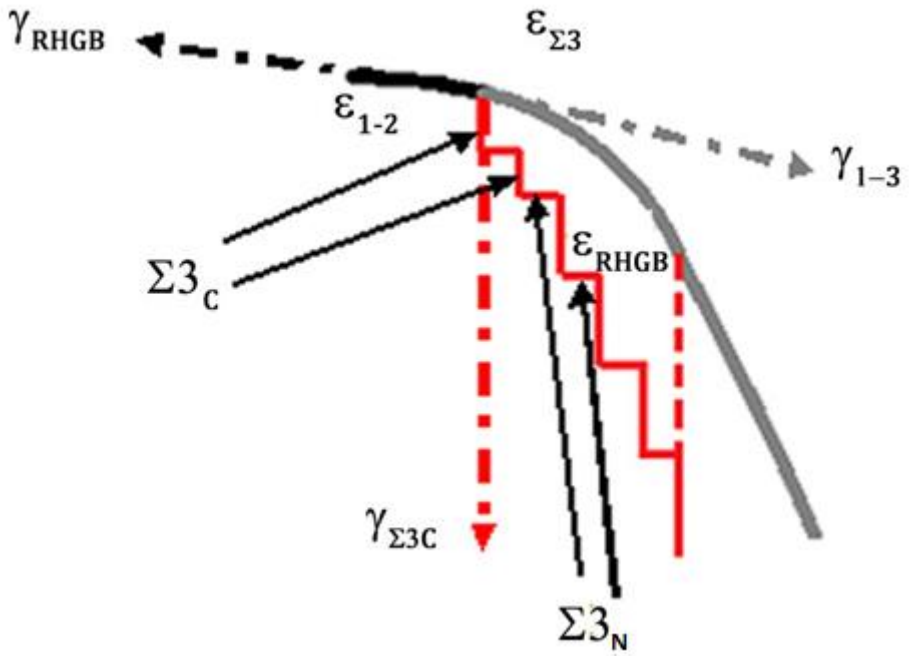


Figure 4.35 Geometry of triple point formed by the dissociation of Σ3 boundary into coherent and incoherent facets.

Figure 4.36 illustrates the formation of a Σ3 with an interface inclination deviated slightly off the coherent {111} plane ($\Sigma3_V$). While the change in geometry due to the non-coherent configuration allows

the Σ3 interface to form a triple point with a suitable geometry to satisfy Young's condition, the increase in boundary energy ($\gamma_{\Sigma 3V} > \gamma_{\Sigma 3C}$) must still be considered,

$$\gamma_{\Sigma 3V} A_{\Sigma 3V} < A_{RHGB}(\gamma_{RHGB} - \gamma_{1-2})$$

Equation 4.11

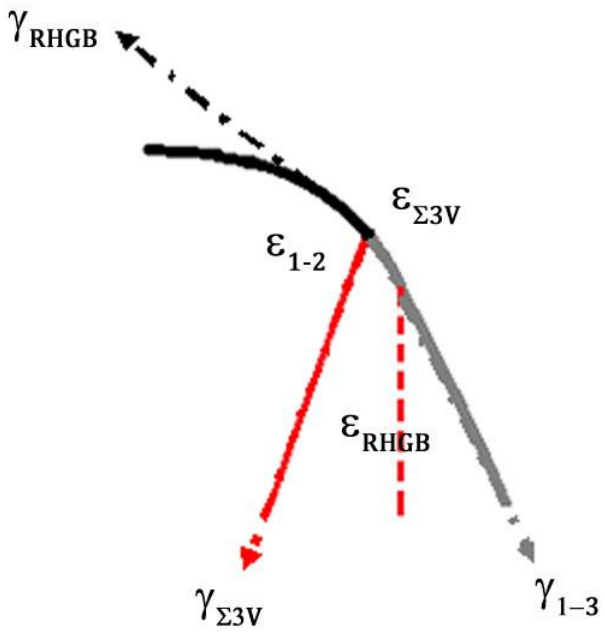


Figure 4.36 Geometry of the triple point formed due to the formation of a $\Sigma 3_V$.

The interface geometry of the Σ3 boundary will be determined by which arrangement provides the most favourable energy condition. Randle [84] showed through grain boundary plane assessment of copper that, after 1 hour at 900°C, less than 15% of the 207 Σ3 boundaries were coherent. Approximately 30% of the 207 Σ3 boundaries had planar indices representing an interface positioned within a 10° tilt around the <110> axis away from the {111} plane. For copper, these asymmetric <110> tilts produce energies up to 10 times larger than that of a coherent twin [63].

After a further 97 hours at 540°C, the number of coherent Σ3 boundaries had increased to 65%, while the asymmetric tilt boundaries decreased to approximately 15%. The result suggests that it is favourable for a Σ3 asymmetric tilt boundary to initially form during recrystallization, and then dissociate into a boundary with an interface comprised of coherent ledges and non-coherent risers when energetically favourable to do so.

Figure 4.37 shows the coherency (λ_A) of the Σ3 interfaces that form part of a Σ3-Σ3-Σ3^n junction compared to those that do not, which are called discrete Σ3s. The results show that the Σ3 interfaces that form these junctions are more likely to have a mixture of coherent and non-coherent boundary segments, while discrete twins were likely to have the majority of their interface identified as coherent. The high λ_A values for individual Σ3 boundaries are indicative of limited faceting along the coherent boundary length, while the opposite is true for the Σ3 boundaries forming junctions. This faceting is observed, Figure 4.38, in an optical image from twin volume Cluster 7 with Σ3s identified in red, Σ9s in blue, and Σ27s in green.

For the case of a discrete Σ3 intersecting a RHGB, the RHGBs can orientate their interface geometries without a significant increase in energy, and thus Young's condition is satisfied with limited faceting along the coherent Σ3 boundary. The opposite is true for junction Σ3 boundaries, where the minimization in boundary energy makes it necessary for a Σ3 to produce incoherent boundary facets due to the energy penalty associated with the reorientation of an entire Σ3 interface.

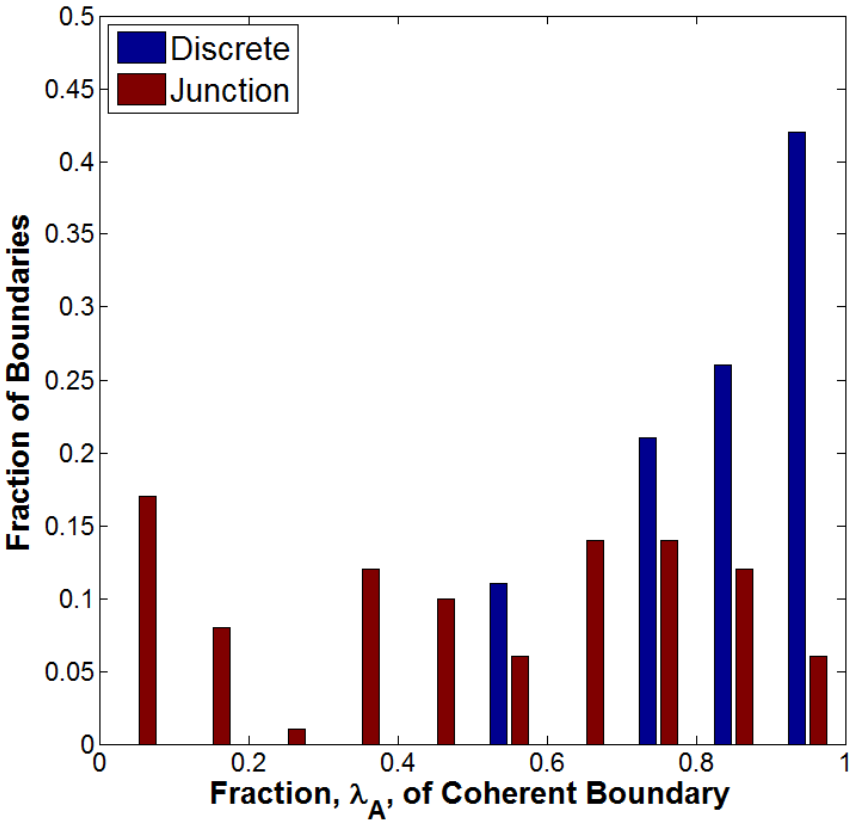


Figure 4.37 Coherency in interface (λ_A) for twin boundaries belonging to a Σ3-Σ3-Σ3^n junction and discrete twin boundaries.

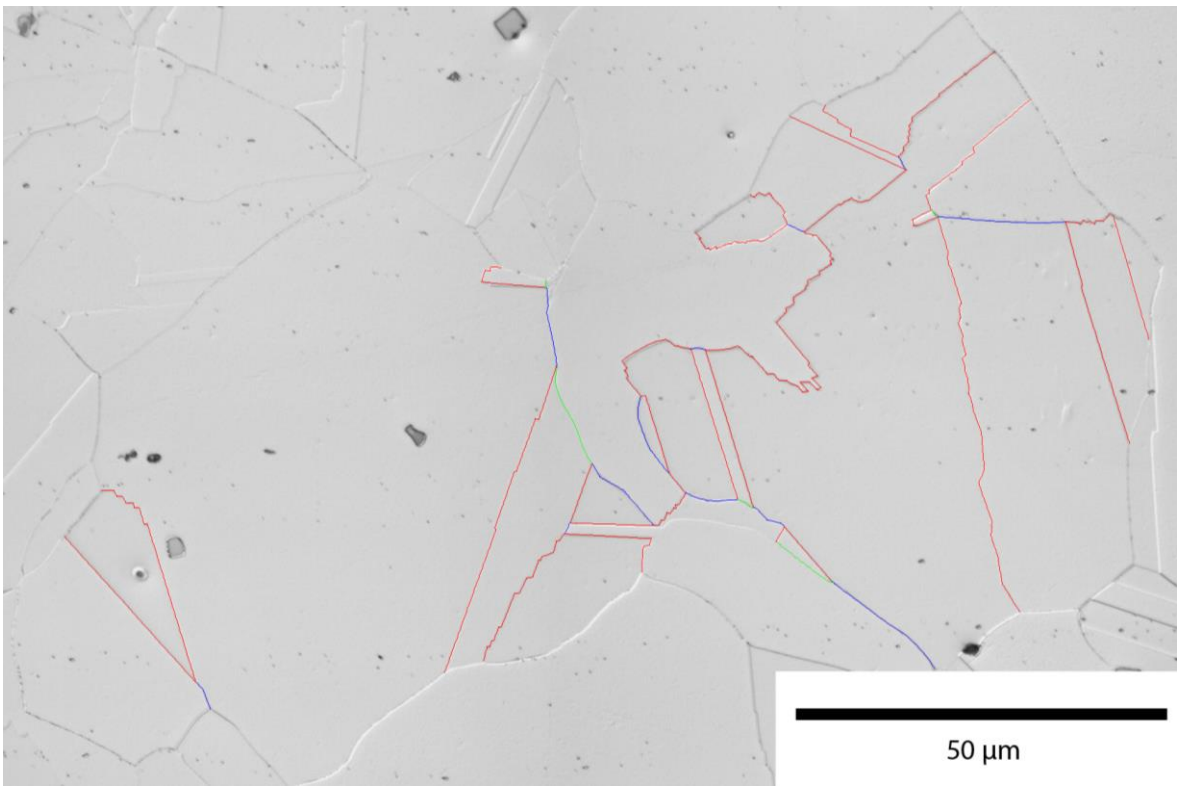


Figure 4.38 Optical section from twin Cluster 7 showing Σ3s in red, Σ9s in blue, and Σ27s in green.

4.4 Summary

Chapter 4 discussed the serial sectioning and 3D reconstruction of Alloy 800H. The reconstructions allowed for analysis of the morphology of Σ3 volumes and the geometries of Σ3 interfaces. The serial sectioning data provided a means to validate trace analysis for the identification of twin boundaries and stereographic corrections in the measurement of grain size statistics.

The measurement of 3D grain size was used to validate the Saltykov stereographic correction for grain size distributions measured in 2D. Saltykov correction increased the number of grains at the upper tail of the distribution. This shift in the distribution increased the average grain size from 51.5µm to 61.0µm, which is within 6% of the average grain size measured from the reconstructed EBSD serial sections of 64.9µm.

In total, seven clusters, containing 70 Σ3 volumes, were reconstructed, producing 138 $\Sigma3^n$ (n = 1-4) interfaces. The 3D morphology of Σ3 volumes was discussed, revealing significant deficiencies in classifying morphologies from 2D sections. Simple Σ3 volumes were able to generate 2D morphologies which were somewhat indicative of the traditional 2D descriptors, although it was shown to be highly dependent on the position of the sectioning plane. Overall, the morphologies represented in 2D provided a poor representation of a volume's true complexity. Intersecting $\Sigma3^n$ boundaries produced 3D morphologies with multiple re-entrant geometries unable to be reliably represented on a single 2D section. The measurement of the interface coherency, λ_A, was used to validate the measurement of coherency on EBSD boundary reconstructions, λ_l, via trace analysis. While discrepancies did exist between λ_A and λ_l the overall trend was similar for both sets of data.

The existence of facets along boundaries was discussed in relation to the current understanding of Σ3 formation. It was suggested that the Σ3 boundaries that formed during recrystallization would initially present interface inclinations described as asymmetric <110> tilts, however, upon further annealing it may have become energetically favourable for the boundary to dissociate into coherent and non-coherent segments. This dissociation created the facetted appearance typical of twin boundaries.

The proportion of coherent interface for each boundary, λ_A, provided a measure of faceting for a twin boundary. It was observed that twin boundaries belonging to Σ3-Σ3-$\Sigma3^n$ junctions were more likely to have lower λ_A values (i.e. more faceting) then those that did not. This faceting was seen as a result of the energy penalty associated with orientating Σ3 interfaces away from a {111} plane, making the dissociation into coherent and non-coherent facets favourable.

Chapter 5 Processing Alloy 800H for Varying Grain Size and Boundary Character Distributions

5.1 Introduction

The purpose of this chapter is to understand how varying strain and annealing temperature affects average grain size, grain size distributions, Σ3 boundary length fraction, and grain boundary network topology for Alloy 800H. The results from this study will be used to produce samples for creep testing.

5.2 Alloy 800H Processing Conditions

53 samples were prepared from as-received (AR) Alloy 800H pipe using combinations of cold-work, 6-80% reduction in thickness (RT), and annealing temperature, 1000-1350˚C. The samples were organised into nine sample sets, Table 5.1 and Table 5.2, to co-ordinate analysis. Table 5.1 summarizes the single-step annealing treatments used for sample sets 1-7. Table 5.2 summarizes the two-step annealing treatments for sample sets 8 and 9. Annealing time was varied to produce a range of average grain sizes.

For sample set 8, step 1 of the annealing treatment was recrystallizing at 1300°C. Step 2 of the annealing treatment was grain coarsening at either 1150°C or 1300°C. For sample set 9, step 1 of the annealing treatment was recrystallizing by slow furnace heating (50°C/hour) from room temperature to 1000°C, followed by a soak for two hours at 1000°C. Step 2 of the annealing treatment was grain coarsening at either 1150°C or 1300°C.

EBSD mapping was used to calculate recrystallization fraction, average grain size, coefficient of variation (width of the grain size distribution), grain boundary character, and grain boundary network topology.

Table 5.1 Single-step annealing treatments for sample sets 1-7. The *Average Grain Size* is given by Equation 3.19, the G*rain Size %Error* is given by Equation 3.20, and the *Coefficient of Variation* describing the width of the grain size distributed grain size measurements is given by Equation 3.23.

	Processing Parameters			**Results**						
Sample Set	*Cold-work (%RT)*	*Annealing Temperature (°C)*	*Annealing Time (min)*	*Average Grain Size (μm)*	*Grain Size %Error (±%)*	*Coefficient of Variation (CV)*	*Σ3 Length Fraction (%)*	*Coherent Σ3 Length Fraction (%)*	*Σ9+Σ27 Length Fraction (%)*	*Recrystallization Fraction (%)*
1	0	1200	0	49.3	9.2	1.026	50.6	28.1	2.1	95.8
			90	82.4	10.8	0.883	50.2	27.1	2.4	99.3
			180	92.5	10.1	0.999	51.1	27.9	2.3	99.1
			240	133.8	12.1	0.925	51.6	23.5	2.3	97.0
2	20	1200	2	41.4	8.2	0.950	49.6	26.9	1.5	95.1
			5	74.6	9.1	0.896	49.5	29.7	1.9	98.9
			10	87.6	10.5	0.873	49.3	28.2	2.0	98.6
			15	101.7	10.2	0.979	51.9	28.1	2.1	98.1
			30	106.1	9.8	0.843	48.9	27.5	1.4	98.3
			60	113.6	12.1	0.986	49.2	28.7	1.0	99.9
			120	123.8	11.9	0.915	50.8	29.2	1.7	97.6
3	20	1225	2	43.4	8.9	0.923	50.3	28.9	2.2	97.9
			5	78.9	9.2	0.934	51.6	28.5	2.1	98.6
			10	98.3	10.3	0.886	51.4	28.6	1.6	98.6
			20	108.0	9.3	0.950	51.6	31.0	2.1	97.6
			30	109.3	9.8	0.898	49.4	23.3	1.5	99.1
			60	125.2	10.1	0.851	50.5	27.0	1.2	97.6
			120	138.8	11.0	0.886	50.7	26.4	2.0	98.9

4	20	1250	2	49.1	8.9	0.901	47.5	26.7	2.0	98.4
			5	99.1	9.1	0.917	50.4	27.7	1.5	97.5
			10	128.3	10.1	0.848	48.5	28.6	1.4	98.4
			20	148.0	9.9	0.849	49.9	28.0	1.1	97.5
			40	155.8	10.1	0.789	49.3	28.1	1.1	99.8
			60	163.4	13.1	0.892	50.0	26.3	1.6	99.2
			120	178.4	12.7	0.860	49.3	27.1	1.0	98.1
5	6	1200	2	54.7	9.9	1.024	54.9	31.7	2.9	99.1
			5	73.4	10.2	0.900	51.0	27.1	3.6	97.8
			15	78.7	11.5	0.833	53.4	29.1	2.7	98.5
			30	81.1	11.0	0.981	57.3	31.7	4.6	98.7
			60	86.5	8.2	0.961	57.1	33.6	4.4	99.6
			120	95.2	8.6	0.867	55.2	31.6	3.7	97.5
			240	95.0	9.9	0.744	48.7	27.8	2.5	96.6
6	60	1200	2	35.8	7.9	0.797	46.2	26.3	1.5	97.2
			10	69.8	8.2	0.833	49.3	30.5	1.7	98.8
			15	71.2	9.3	0.872	48.3	27.0	1.7	95.1
			30	76.0	8.7	0.817	47.0	25.8	1.5	97.3
			60	87.3	8.9	0.884	49.0	29.6	1.7	97.5
			120	87.2	10.3	0.845	47.3	27.7	1.8	96.2
			240	104.9	11.14	0.747	48.7	26.4	1.5	97.9
7	80	1350	1.5	72.7	9.8	0.701	41.9	23.8	1.8	98.3
			5	96.3	9.9	0.714	42.6	27.4	1.1	98.4
			10	112.3	10.9	0.714	44.9	24.6	1.5	96.7
			20	112.6	11.2	0.627	40.5	24.1	0.9	97.0

Table 5.2 Two-step annealing treatments for sample sets 8 and 9. The *Average Grain Size* is given by Equation 3.19, the G*rain Size %Error* is given by Equation 3.20, and the *Coefficient of Variation* describing the width of the grain size distributed grain size measurements is given by Equation 3.23.

	Processing Parameters					**Results**						
	Step 1			**Step 2**								
Sample Set	*Cold-work (%RT)*	*Annealing Temperature (°C)*	*Time (min)*	*Annealing Temperature (°C)*	*Time (min)*	*Average Grain Size (μm)*	*Grain Size %Error (±%)*	*Coefficient of Variation (CV)*	*Σ3 Length Fraction (%)*	*CoherentΣ3 Length Fraction (%)*	*Σ9+Σ27 Length Fraction (%)*	*Recrystallization Fraction (%)*
8	80	1300	0.5			32.1	8.1	0.770	44.6	27.0	1.3	96.9
				1150	15	44.6	7.9	0.855	46.5	28.3	1.0	98.0
					30	45.6	10.1	0.858	47.0	27.6	1.4	97.3
				1300	1	47.9	10.5	0.833	45.5	24.1	1.5	97.8
					2	87.3	9.8	0.889	44.5	25.3	2.3	97.0
9	20	1000	1320			24.0	7.8	0.960	57.4	30.1	7.2	98.4
				1150	15	38.0	9.8	0.991	54.6	31.3	3.7	97.8
					30	39.5	10.1	0.987	51.7	29.3	4.2	97.8
				1300	0.5	29.2	9.3	1.101	55.4	29.8	7.1	95.4
					1	51.3	9.4	0.912	50.2	27.2	2.3	95.7

5.3 Results and Discussion

5.3.1 Alloy 800H Microstructures

Figure 5.1 shows examples of Alloy 800H grain boundary networks for four samples with varied processing histories. 20%RT/1200°C/10min denotes a sample cold-worked to a 20% reduction in thickness, before being annealed at 1200°C for 10 minutes. Figures 5.1(a) and 5.1(b) show the grain boundary networks for the AR material and 20%RT/1200°C/10min sample, respectively. Both samples produced similar Σ3 length fractions, ≈50%, and Σ9 + Σ27 length fractions, ≈2%. Figure 5.1(d) shows the grain boundary network for the 80%RT/1350°C/1.5min sample. The sample produced Σ3 and Σ9 + Σ27 length fractions lower than those observed in the microstructures shown in Figures 5.1(a) and 5.1(b), 41.9% and 1.8%, respectively. Figure 5.1(c) shows the sample strained to 20% RT and annealed at 1000°C for 22 hours. The Σ3 and Σ9 + Σ27 length fractions were higher than any other sample produced in the study, 57.4% and 7.2%, respectively.

Figure 5.1(d) shows the Σ3 boundaries appearing as isolated features within clearly defined grains, resulting in minimal breakup of the random high-angle grain boundary (RHGB) network. This observation was confirmed by the triple junction distributions, Figure 5.2, where a low frequency of $2\Sigma3^n$ and $3\Sigma3^n$ junctions was observed, 4% and 3%, respectively. Grain boundary networks shown in Figures 5.1(a) and 5.1(b) appear similar to that of Figure 5.1(d), although the additional Σ9 and Σ27 boundaries resulted in a higher frequency of $2\Sigma3^n$ and $3\Sigma3^n$ junctions, ≈5% and ≈8%, respectively. The RHGB network observed in Figure 5.1(c) is highly disconnected due to the formation of a large number of $2\Sigma3^n$ and $3\Sigma3^n$ junctions, 6% and 24%, respectively. The Σ3 boundaries in Figure 5.1(c) are observed to be an integral part of the grain boundary network, display a curved appearance, and are more likely to be non-coherent and mobile.

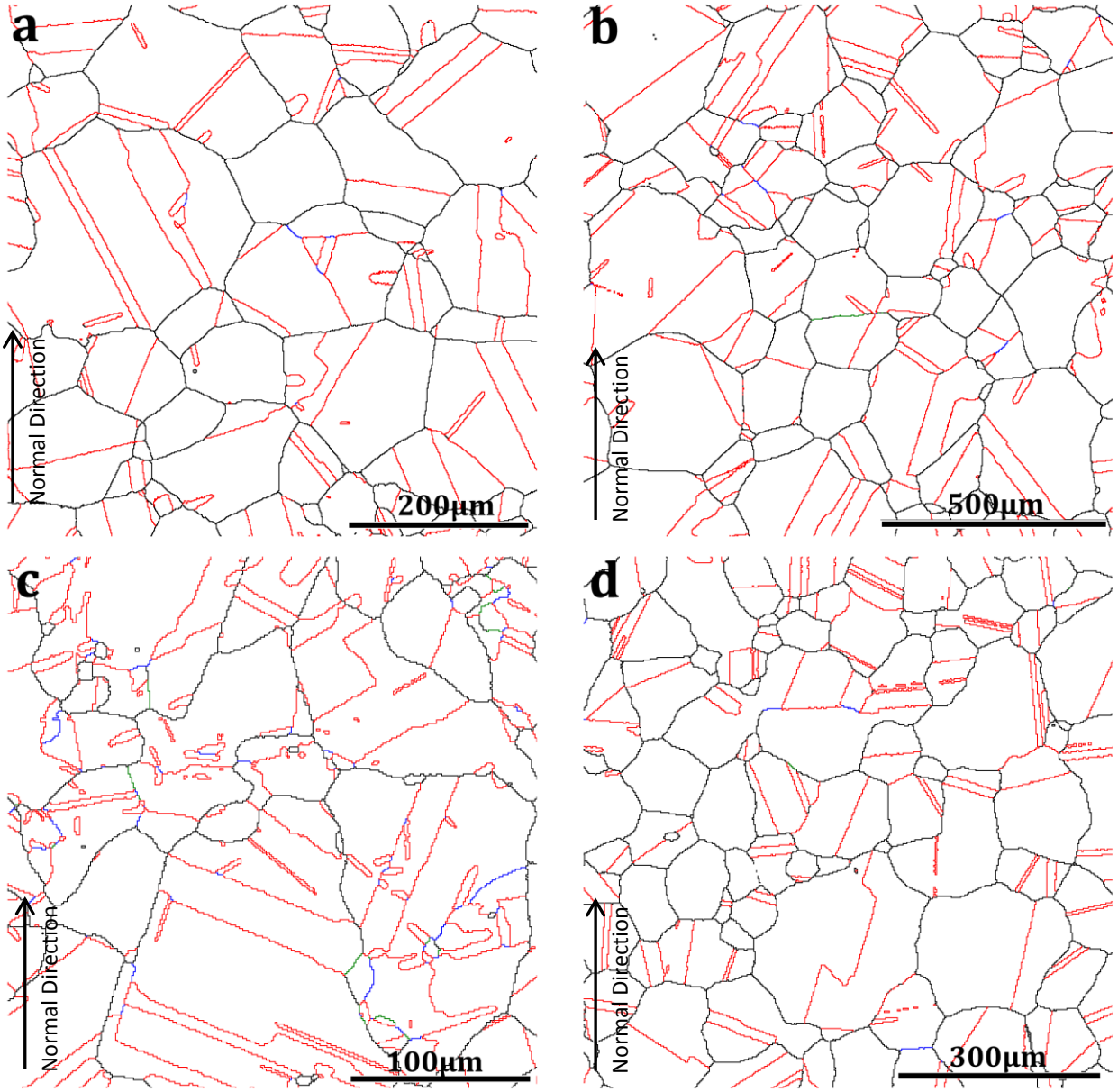


Figure 5.1 EBSD grain boundary maps. Red = Σ3, Blue = Σ9, Green = Σ27, Black = Other High Angle Grain Boundaries. (a) AR, (b) 20%RT/1200°C/10min, (c) 20%RT/1000°C/22hours, (d) 80%RT/1350°C/1.5min.

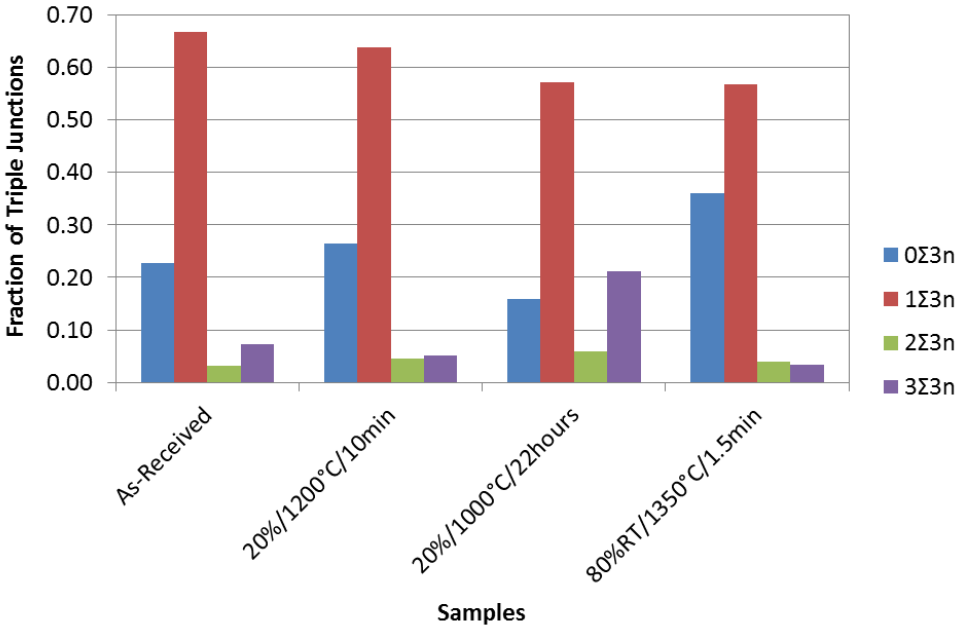


Figure 5.2 Triple junction distributions for the four samples shown in Figure 5.1.

5.3.2 Minimum Deformation for Recrystallization

A minimum amount of strain is required to initiate the recrystallization of grains and result in the formation of a new grain boundary network. Kumar [109] noted that while large strains produced walls of dislocation, an ideal environment for recrystallization, small strains did not. At low strains, the distribution of dislocations within the grains of the original microstructure was more homogenous, although differences in dislocation densities existed between grains. This strain energy gradient between grains served as a driving force for the migration of existing boundaries at elevated temperatures.

Samples from sets 2, 3, and 4 were strained to 20% RT, and after 2 minutes annealing produced average grain sizes of 41.4µm, 43.4µm, and 49.1µm, respectively. The average grain sizes were less than the 49.3µm measured for the AR material, an indication that the original microstructure has been replaced by recrystallization.

Samples from set 5 were strained to 6% RT, producing a grain size of 54.7µm after 2 minutes annealing at 1200°C. The average grain size, 54.7µm, is greater than that average grain size for the AR material, 49.3µm. The result indicates that 6% strain was not sufficient to initiate recrystallization processes, but rather modify the existing AR microstructure by promoting the migration of the existing grain boundaries.

5.3.3 Recrystallization Fraction

Figure 5.3 shows an example of a recrystallization fraction EBSD map for a 20%RT/1200°C/2min sample. The colour represents the grain orientation spread (GOS) of each grain. Additionally, a plot representing the area fraction of grains with GOS is shown in Figure 5.3. All samples in the study were identified as fully recrystallized, i.e. over 95% of the area fraction was identified as recrystallized (less than 4° GOS).

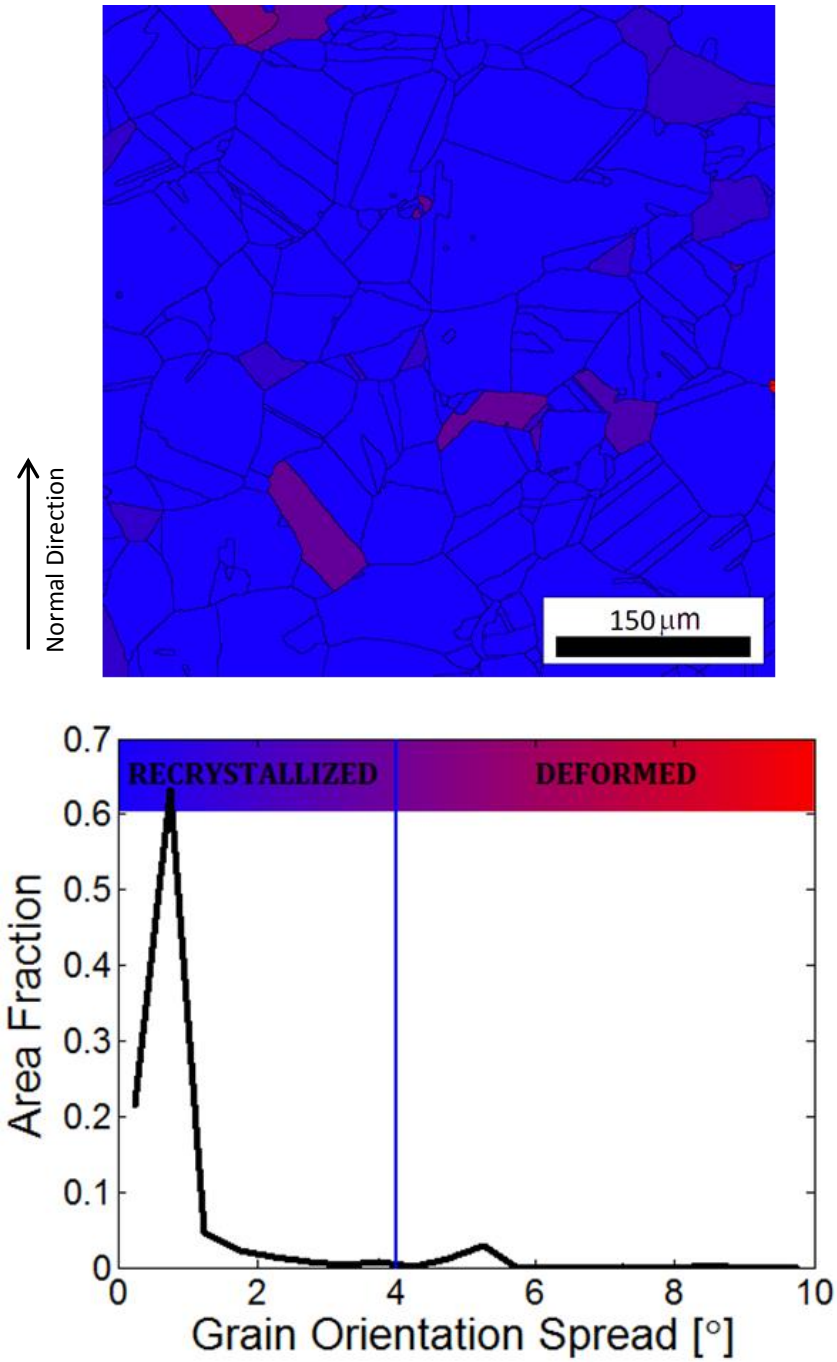


Figure 5.3 EBSD map indicating the grains defined as recrystallized (blue) or deformed (red) for the 20%/1200°C/2min sample.

5.3.4 Recrystallized Grain Size

The average grain size at the completion of recrystallization is proportional to the ratio $G/\dot{N}$, where *G* is the growth rate of recrystallizing grains, and $\dot{N}$ is the rate at which grains nucleate in the deformed microstructure. Both *G* and $\dot{N}$ increase with an increase in strain and annealing temperature.

For samples sets 2, 3, and 4, all strained to 20% RT, the grain size after 2 minutes annealing increased with increasing temperature: 41.4µm 43.4µm, and 49.1µm. The average grain sizes indicate that the increase in annealing temperature from 1200°C to 1250°C had a greater influence on the growth rate of new grains than the nucleation rate, i.e., an increase in $G/\dot{N}$.

The sample strained to 20% RT and annealed at 1000°C for 22 hours produced the smallest average grain size of all the samples, 24.0µm. The slow (50°C/hour) furnace heating rate before reaching 1000°C promoted recovery within the deformed microstructure. The rearrangement of dislocations due to recovery not only provided ideal sites for the nucleation of new grains, but also reduced strain energy, a driving force for growth. The result is a decrease in *G,* combined with an increase in $\dot{N}$, i.e. a smaller average grain size.

The average grain size of the sample strained to 60% RT and annealed at 1200°C for 2 minutes, 35.8µm, is lower than that of the sample strained to 20% RT and annealed at the same temperature, 41.4µm. The increase in strain energy not only provided a driving force for an increased growth rate, but also the heavily deformed microstructure offered a greater number of sites for the nucleation of new grains. The decrease in grain size indicates that an increase in strain from 20 to 60% RT results in a decrease in $G/\dot{N}$, i.e. the nucleation of new grains due to the deformed microstructure is greater than the increase in growth rate.

The sample strained to 80% RT and annealed at 1350°C showed an average grain size of 72.7µm after 2 minutes. It is difficult to determine whether the average grain size was a result of grain coarsening following recrystallization. It is reasonable to conclude that the high annealing temperature would result in the largest growth rates of all the samples.

5.3.5 Width of the Grain Size Distribution – Coefficient of Variation

During recrystallization, nucleated grains will experience varied growth rates, influenced by their orientation relationship with the surrounding microstructure and variations in strain energy within the deformed microstructure [29]. The time at which a grain nucleus forms, their variations in growth rates, and the impingement onto adjacent recrystallizing grains, will result in microstructures with a distribution of grain sizes. The average coefficient of variation for the grain size distributions for the individual sample sets are shown in Figure 5.4. The error bars give the range of coefficient variations for the sample set.

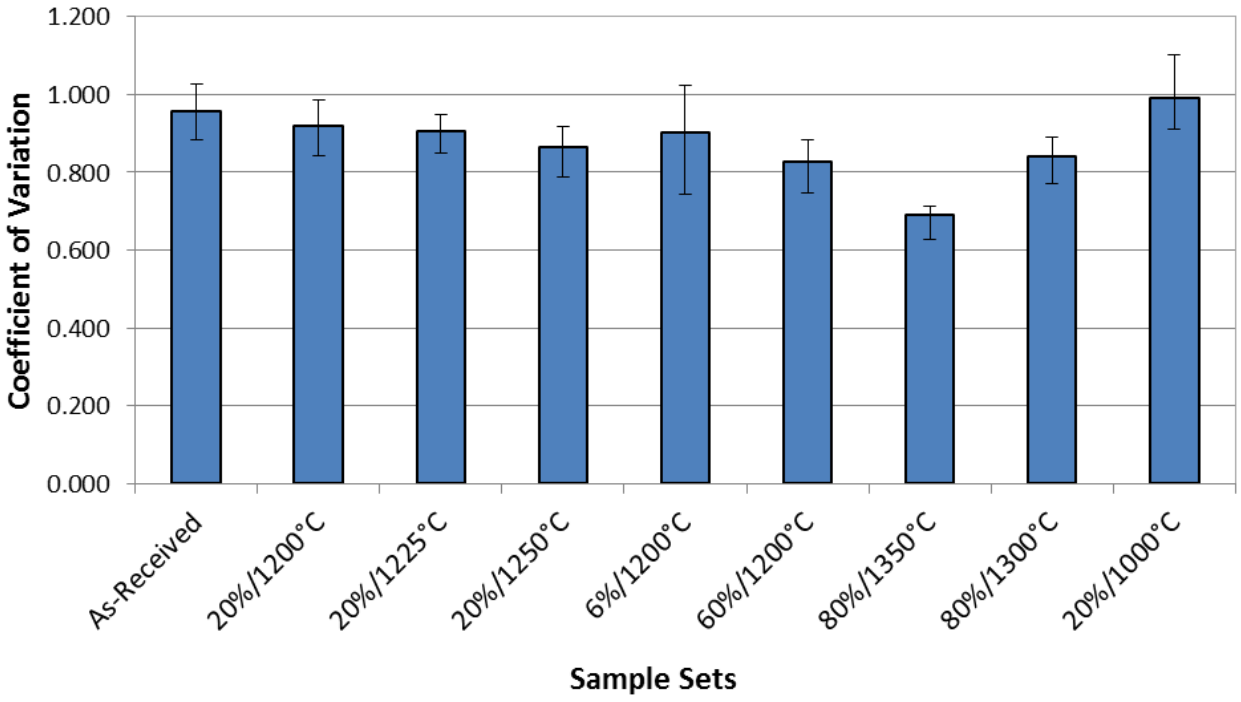


Figure 5.4 Average coefficient of variation for all sample sets.

The results show that the samples from sets 1 to 5 have similar coefficient of variations. There is a minimal decrease in the coefficient of variation as annealing temperature increases, 0.920 and 0.865 for 1200°C and 1250°C, respectively. A more noticeable decrease, 0.920 to 0.828, occurs due an increase in the amount of deformation, 20% to 60% RT. The formation of narrow grain size distributions after increasing the degree of deformation has been noted previously in Ti and Al [38, 147].

The smallest and largest coefficient of variation, 0.689 and 0.990, belong to sample sets 7 and 9, respectively. The processing conditions for sample set 7, 80% RT followed by annealing at 1350°C, results in the fastest recrystallization times. Conversely, the slowest recrystallization times are due to

the low temperature anneal performed on sample set 9. The results suggest that an increase in recrystallization rate will create samples with narrow grain size distributions.

5.3.6 Grain Coarsening

Further annealing after recrystallization promotes grain coarsening. Equation 2.11 describes the increase in average grain size to $\overline{d}$ from an initial grain size $\overline{d_0}$ as a function of the annealing temperature, *T,* after time, *t*. Sample sets 2, 3, and 4 reflect this increase in average grain size in relation to increased annealing temperature, Figure 5.5. After 10 minutes at temperature, the average grain sizes for sample sets 2, 3, and 4, were 73.8µm, 98.3µm, and 128.3µm, respectively.

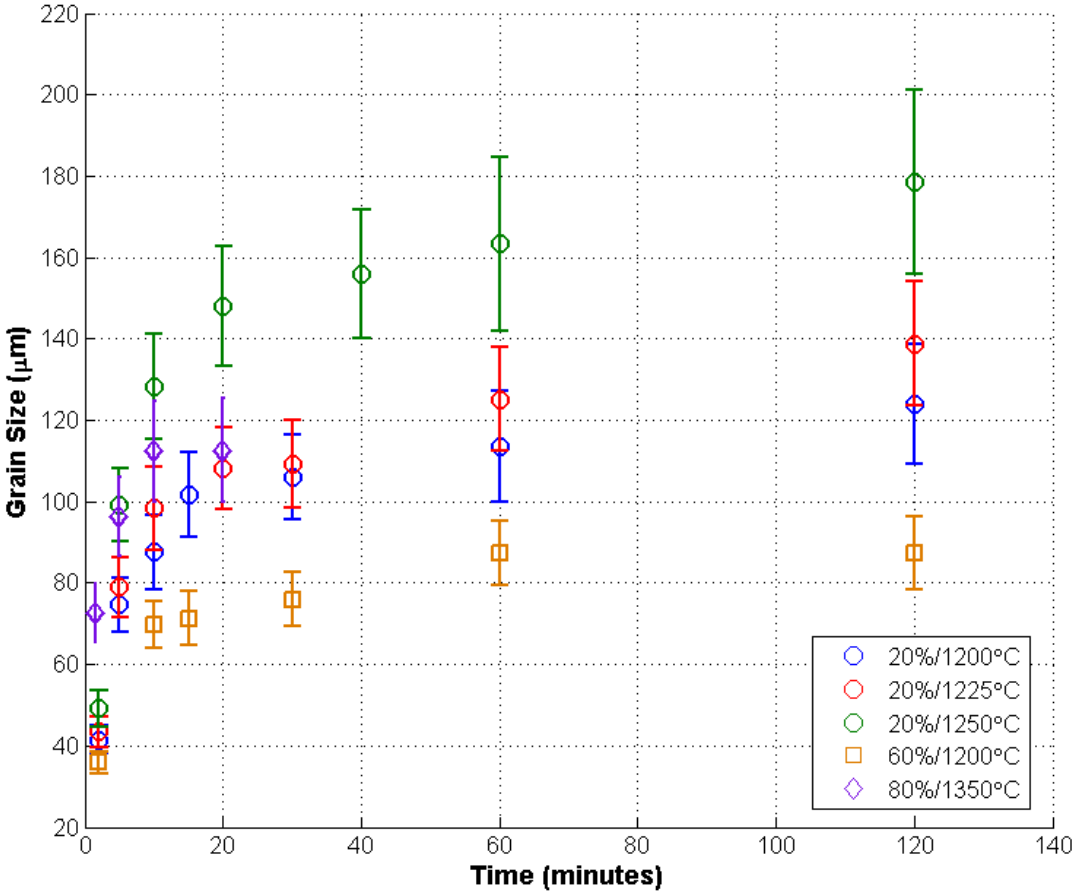


Figure 5.5 Average grain size after annealing between 2 and 120 minutes for sample sets 2, 3, 4, 6, and 7.

Fitting Equation 2.11 to the average grain size data of sample sets 2, 3, and 4, values for the activation energy for boundary growth, *Q*, grain growth exponent, *n*, and fitting constant, k_0, were calculated. Typically, the activation energy for the transfer of atoms across a grain boundary should be about half that of self-diffusion [148]. For the current data, the activation energy for boundary motion was determined to be 125kJmol^{-1}, approximately half the activation energy for the diffusion of iron in Alloy 800H (259.6kJmol^{-1} [149]). The grain growth exponent, n = 5.85, was calculated for the temperature range with a fitting constant of 4.82x10^5. Figure 5.6 shows the average grain size measured from the samples, with curves produced using n = 5.85 and a fitting constant of 4.82x10^5.

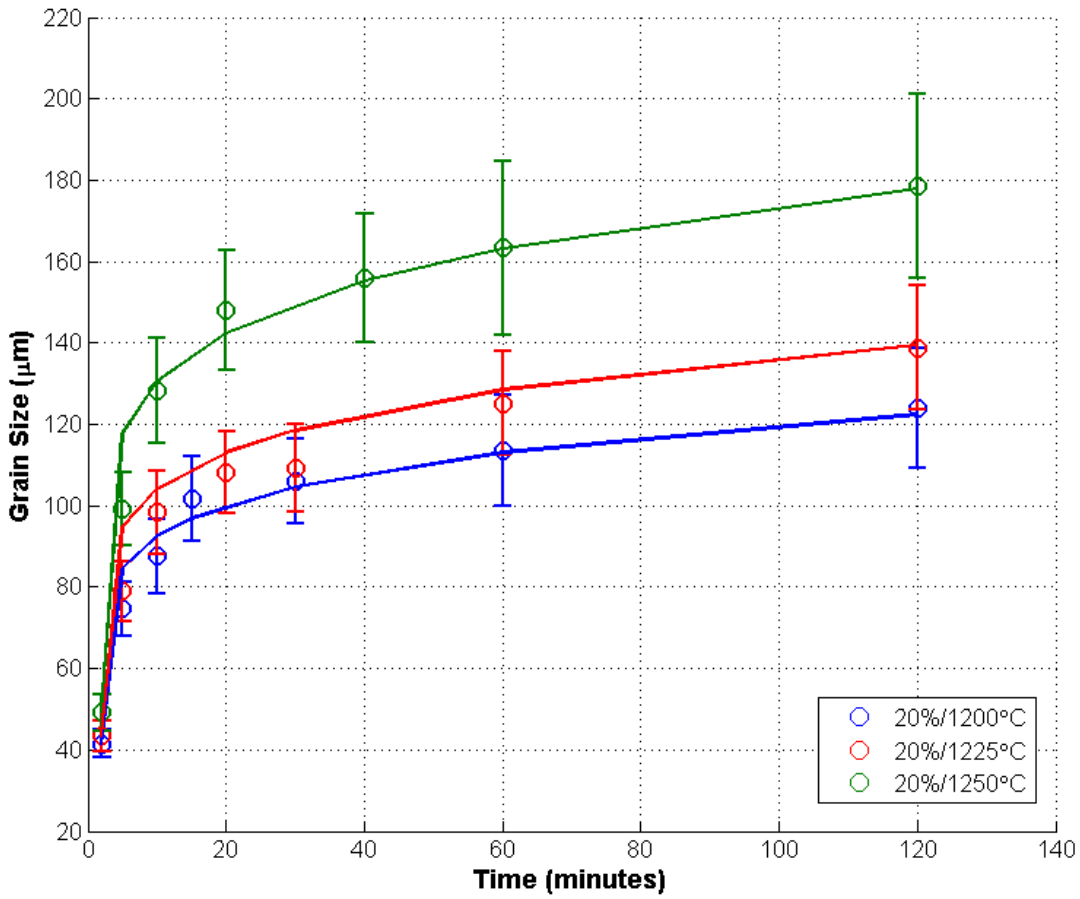


Figure 5.6 Equation 2.11, power law for grain growth, fitted to sample sets 2, 3, and 4.

The major limitation with Equation 2.11 is that there is no consideration given to the grain size distribution of the recrystallized microstructure. Due to the driving force for larger grains to grow at the expense of smaller grains, samples with the widest grain size distribution will have the fastest increases in average grain size.

Sample set 6 was annealed at the same temperature as sample set 2 (1200°C), although samples show considerably different average grain sizes after the same anneal times. For example, after 120 minutes, the average grain size of the sample from set 6 was 87.2µm compared to 123.8µm for set 2, even though the initial average grain sizes of the two sets were similar - 35.8µm and 41.4µm, respectively. Figure 5.4 shows that samples from set 6 (60%/1200°C) had tighter grain size distributions compared to those from set 2. The tighter grain size distribution is due to fewer small grains, resulting in a microstructure stable against grain growth.

5.3.7 Σ3 Boundary Formation

The process of Σ3 formation is often described as the dissociation of a migrating high angle grain boundary into two boundaries with an energy sum per unit area less than that of the original boundary [91]. Aust and Rutter [150] identified that the boundaries produced when a high angle boundary dissociates can typically be characterised by a CSL misorientation.

Goodhew [99] was able to identify the dissociation of high angle boundaries in small (<1µm) recrystallizing grains in gold. Goodhew identified three dissociation reactions:

$$\Sigma 9 \rightarrow \Sigma 3 + \Sigma 3$$

$$\Sigma 11 \rightarrow \Sigma 3 + \Sigma 33$$

$$\Sigma 99 \rightarrow \Sigma 3 + \Sigma 33$$

Goodhew suggested that a Σ33 has a lower energy than a Σ11 because a Σ33 was observed to not dissociate into a Σ11 and Σ3, even though it is geometrically possible to do so.

Kumar et al [109] identified several occurrences of grain boundary dissociation during strain annealing of lightly deformed copper, Inconel 600, Inconel 690, and Hastelloy C-22.

$$\Sigma 27a \rightarrow \Sigma 9 + \Sigma 3$$

$$\Sigma 51a \rightarrow \Sigma 17b + \Sigma 3$$

$$\Sigma 87b \rightarrow \Sigma 3 + \Sigma 29b$$

Both Jones [103] and Wilbrandt [151] suggested that the formation of Σ3 boundaries promoted the growth of recrystallizing grains. Wilbrandt found that those recrystallizing grains with boundary misorientation of 40°/<111> provided the ideal condition for growth. Several researchers [58, 59, 61,

104, 152] reported that the formation of Σ3 boundaries at static boundaries provided conditions favourable to resume migration.

In the current study, the formation of Σ3 boundaries will be discussed in terms of boundary dissociation, regardless of whether boundary migration is due to recrystallization, or by the migration of existing grain boundaries, i.e. strain annealing.

5.3.8 Σ3 Boundary Formation through Grain Coarsening

Burke [89] originally suggested that Σ3 boundaries may form without the presence of strain, that is, during grain coarsening, after recrystallization. Results from the sample sets 1 to 7 (Table 5.1) suggest that the majority of Σ3 boundaries form during recrystallization. While fluctuations in Σ3 length fractions occur, there seems to be no definitive upwards trend to support twin formation during grain growth.

Sample sets 8 and 9 were designed to confirm that no appreciable Σ3 formation occurs during grain growth. Samples from set 8, Figure 5.7, were recrystallized at 1300°C after 80% RT in an attempt to restrict the formation of Σ3 boundaries. Two different temperatures, 1150°C and 1300°C, were selected to provide contrasting grain coarsening rates upon further annealing. However, there was no substantial difference in Σ3 length fraction between all samples in set 8.

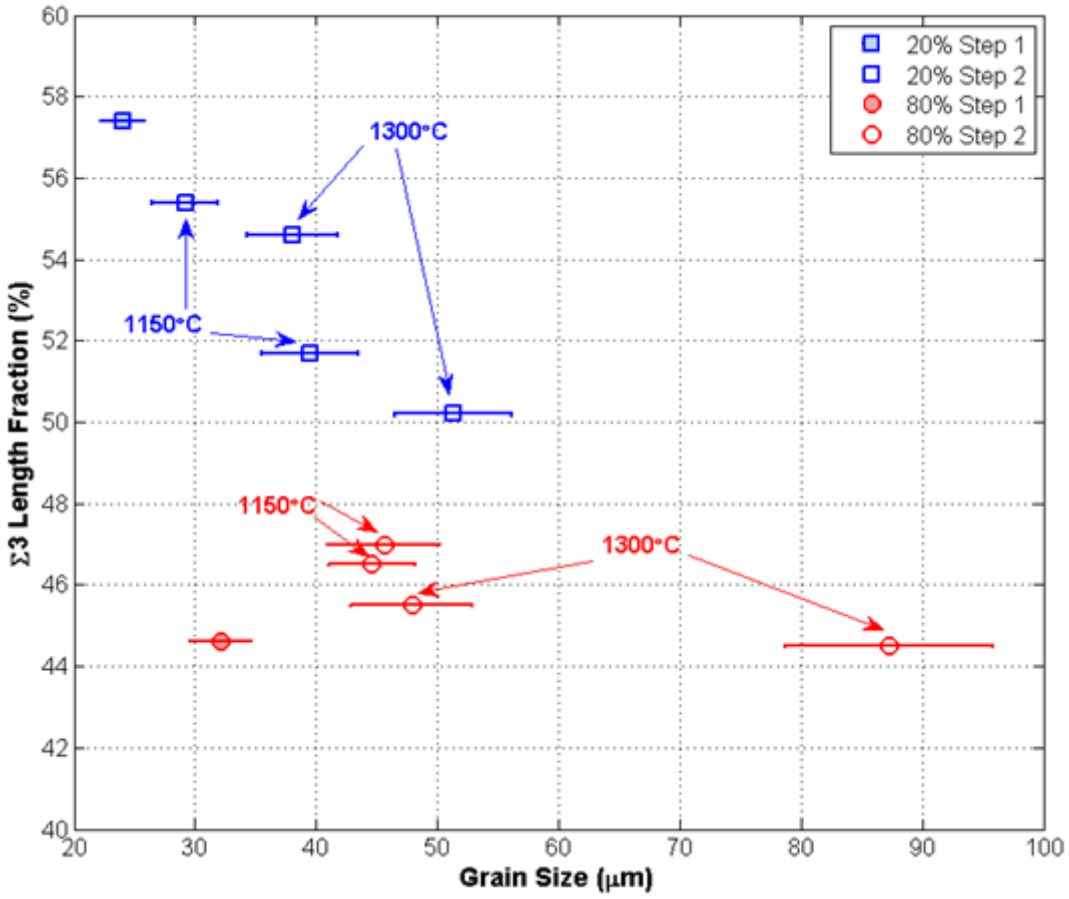


Figure 5.7 Σ3 length fractions for samples of varying grain size from sets 8 and 9. Indicated temperatures are those used for further annealing.

Samples from set 9 were recrystallized at 1000°C after 20% RT, and produced the largest length fraction of Σ3 boundaries of all the samples prepared, 57.4%. Further annealing at 1150°C and 1300°C produced samples with Σ3 length fractions lower than after the initial treatment. This result indicates that grain growth is in fact removing Σ3 boundaries from the microstructure. The removal of Σ3 boundaries from the microstructure during grain growth was also noted by Grube and Rouze [153], affirming the current view that the formation mechanisms for Σ3 boundaries are only operative in the presence of strain energy.

5.3.9 Effect of Temperature and Strain on Σ3 Formation

Figure 5.8 shows the average Σ3 lengths fractions for sample sets 1 to 7. Samples from sets 2, 3, and 4 were all strained to 20% RT, followed by annealing at 1200°C, 1225°C, and 1250°C, respectively. The average Σ3 length fractions were approximately 50% for all three sample sets, suggesting that the current range of annealing temperatures had little effect on the formation of Σ3 boundaries. The limited

effect of temperature on the formation of Σ3 boundaries was also noted by Gleiter [90], Li [95], and Pande [96].

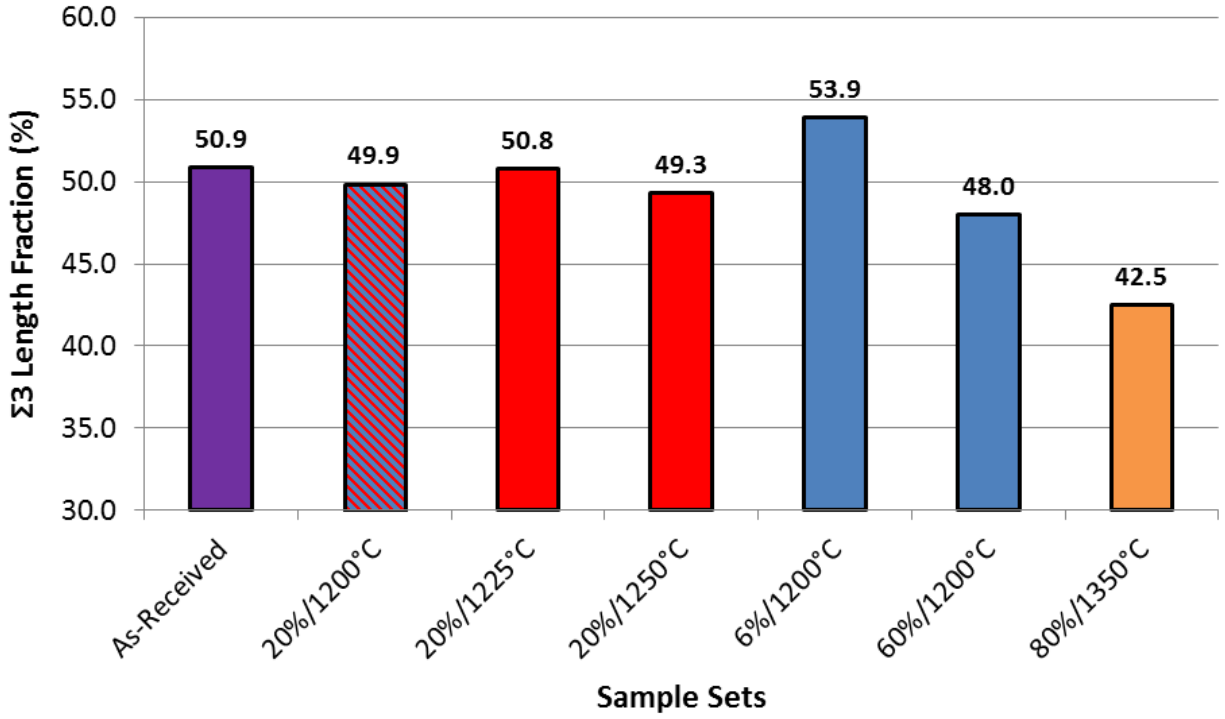


Figure 5.8 Average Σ3 length fractions for sample sets for various processing conditions.

Figure 5.8 also shows the effect of deformation on the Σ3 length fractions for samples strained by 6% (sample set 5), 20% (sample set 2), and 60% (sample set 6) RT prior to annealing at 1200°C. Samples from set 6 (60%/1200°C) typically displayed the lowest overall Σ3 length fractions from all samples in sets 1 to 6 (48%), while samples from set 5 (6%/1200°C) displayed some of the largest Σ3 length fractions (53.9%). The same result has been observed in studies in cold wire drawing of copper, where a decrease in Σ3 length fraction was observed, 47% to 40%, for an area reduction of 52% and 94%, respectively [105]. Kumar [106] identified a decrease in Σ3 length fraction from 34.5% to 28.2% for 304L cold-rolled to a thickness reduction of 60% and 80%, respectively, and annealed at 1000°C.

The decrease in Σ3 length fraction with increasing strain is the result of an increase in driving force for the nucleation and growth of new grains during recrystallization. The energy available due to the increase in strain promotes the migration of high angle boundaries, suppressing the requirement for the formation of Σ3 boundaries. This idea is confirmed by the Σ3 length fractions measured from samples of set 7 (80%RT/1350°C), shown in Figure 5.8. The large strains and high temperature promotes the fast recrystallization of grains, thus producing the lowest Σ3 length fractions observed in the investigation, 42.5%.

Samples from set 5 were strained to 6% RT, sufficiently low to promote strain annealing in lieu of recrystallization at 1200°C. The gradient in strain energy across existing boundaries provides the driving force for migration. Just as in the case of recrystallization, the migration of the boundaries is assisted by dissociation into a Σ3 and a boundary with increased mobility.

Unlike recrystallization, strain annealing promotes the migration and dissociation of $\Sigma 3^n$ boundaries already present in the microstructure ($\Sigma 27a \rightarrow \Sigma 9 + \Sigma 3$). The dissociation of the $\Sigma 3^n$ boundary types promotes the breakup of the RHGB network, and continued migration leads to encounters with other $\Sigma 3^n$ boundaries and the formation of new $\Sigma 3^n$ boundaries by the mechanisms proposed by Randle [116]. Strain annealing is employed in the development of GBE microstructures.

5.3.10 Σ3 Boundary Coherency

In studies investigating Σ3 boundary populations, there is typically no distinction made between coherent (twins) and non-coherent Σ3 boundaries. In the case of copper, the energy ratio between a coherent twin and a non-coherent Σ3 can be as much 50 times [63], while boundary diffusivity may differ by up to a factor of 10 [154]. These are important differences to consider, particularly when correlating material performance to grain boundary character. The coherent twin length fractions for all sample sets are displayed in Figures 5.9 and 5.10. Typically, between 55 and 60% of Σ3 boundary length was identified as coherent.

Figure 5.9 shows that the coherent twin length fraction is essentially constant (≈28%) for sample sets 1, 2, 3, 4, and 6. The length fraction of sample set 5 was slightly higher (30.4%), a result consistent with having approximately 10% more Σ3 boundaries. Likewise, sample set 7 had a coherent twin length fraction slightly lower, 25%, again consistent with having an overall lower fraction of Σ3 boundaries.

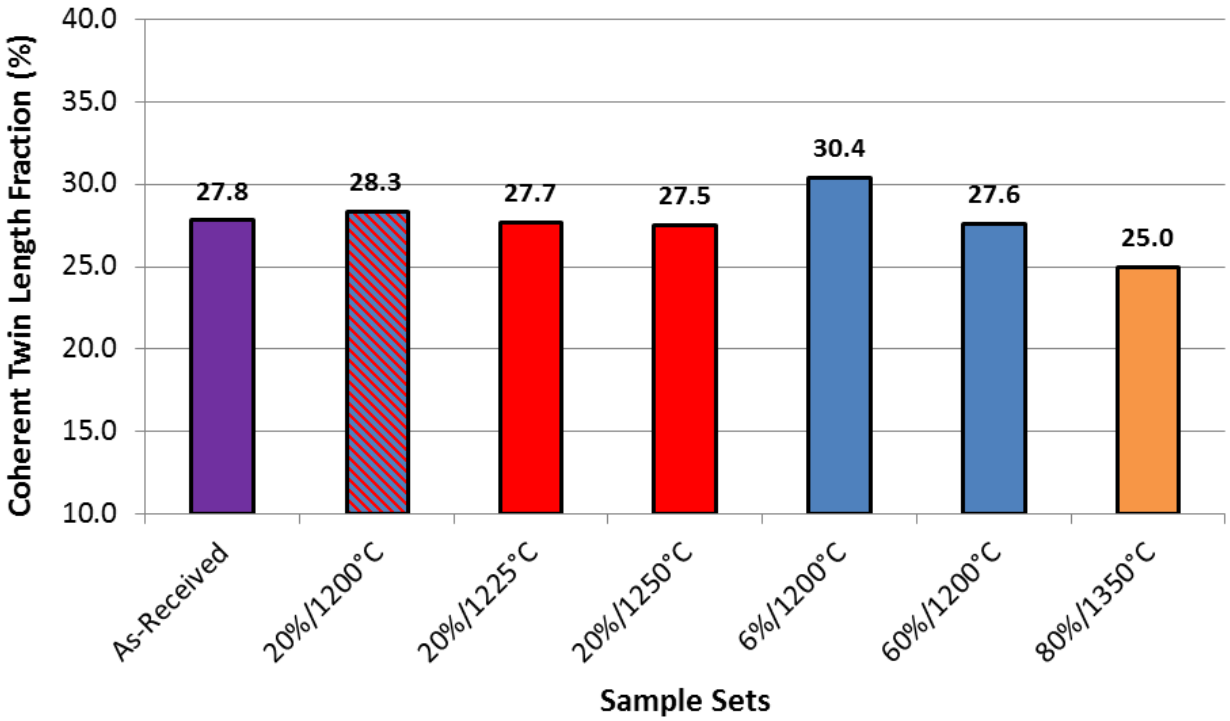


Figure 5.9 Coherent twin boundary length fractions for sample set 1 to 7.

Figure 5.10 shows the coherent twin length fractions for the two step processed sample sets 8 and 9. Sample set 8 (80% strain) showed a similar trend to sets 1 to 7, in that approximately 55% of the Σ3 boundary length was coherent. The coherent twin length fractions of the samples after processing step 1 were approximately equal for both sample sets 8 and 9, 27.0% and 27.3%, respectively. This result is in contrast to that presented in Figure 5.7, showing a difference in the Σ3 length fraction of 12.8% between the same two samples. The result indicates that the additional Σ3 boundary length produced by the low temperature recrystallization was non-coherent. Figure 5.7 also showed the annihilation of Σ3 boundaries during grain coarsening, while Figure 5.10 shows the length fraction of coherent twins increasing. The result suggests that the Σ3 boundary length removed during grain coarsening was non-coherent. The non-coherent Σ3 boundaries are more likely to be mobile during annealing.

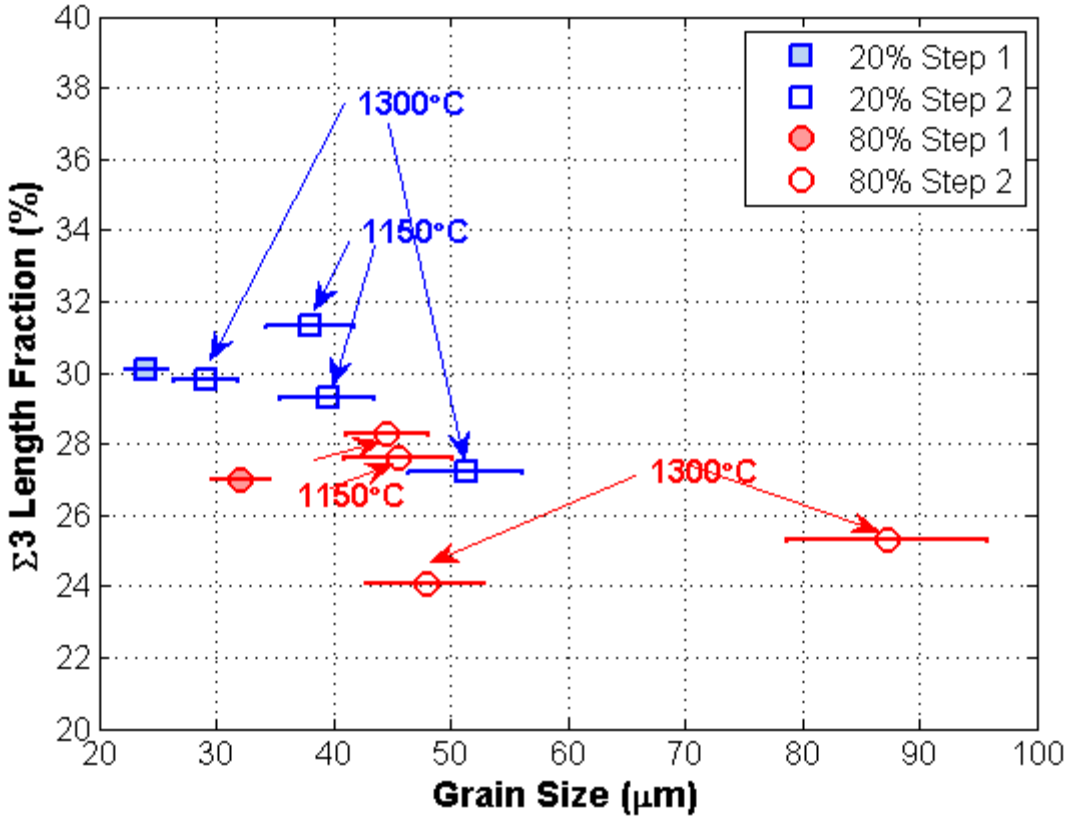


Figure 5.10 Coherent twin boundary length fractions for samples of varying grain size from sets 8 and 9.

5.3.11 Grain Boundary Network Connectivity

Figure 5.11 shows the distribution of triple junctions with 0, 1, 2, and 3Σ3^n (n≤3) boundaries. Analysing the distribution of triple junctions by this method provides an indication of the connectivity of the RHGB network. Sample sets 2, 3, 4, and 6 represent processing conditions that in the current study are considered to promote moderate recrystallization rates. The frequency of triple junctions containing 0Σ3^n, ≈26%, is typical of all samples from sets 2, 3, 4, and 6, and is consistent with that seen in the AR condition. The number of triple junctions containing 3Σ3^n is again consistent with that measured in the AR sample, ≈7%.

By accelerating the recrystallizing rate in sample set 7, the length fraction of Σ3 boundaries was reduced from approximately 50% to 42.5%. This resulted in 36% of triple junctions containing 0Σ3^n boundaries and only 3% of triple junctions containing 3Σ3^n boundaries. The result indicates that many of the Σ3

boundaries do not form an integral part of the grain boundary network, but rather Σ3 boundaries are observed as isolated features within grains.

Due to the increase in Σ3 length fraction in samples from set 5 (6%/1200°C), the RHGB network displays greater discontinuity compared to any sample from sets 1 to 8. This is shown in Figure 5.11 by a decrease in triple junctions containing $0\Sigma3^n$ boundaries (21%), and an increase in those containing $3\Sigma3^n$ boundaries (13%). The greatest break-up of the RHGB network was achieved by the sample annealed at 1000°C. The fraction of $0\Sigma3^n$ boundaries was 13%, and triple junctions containing $3\Sigma3^n$ boundaries increased to 24%.

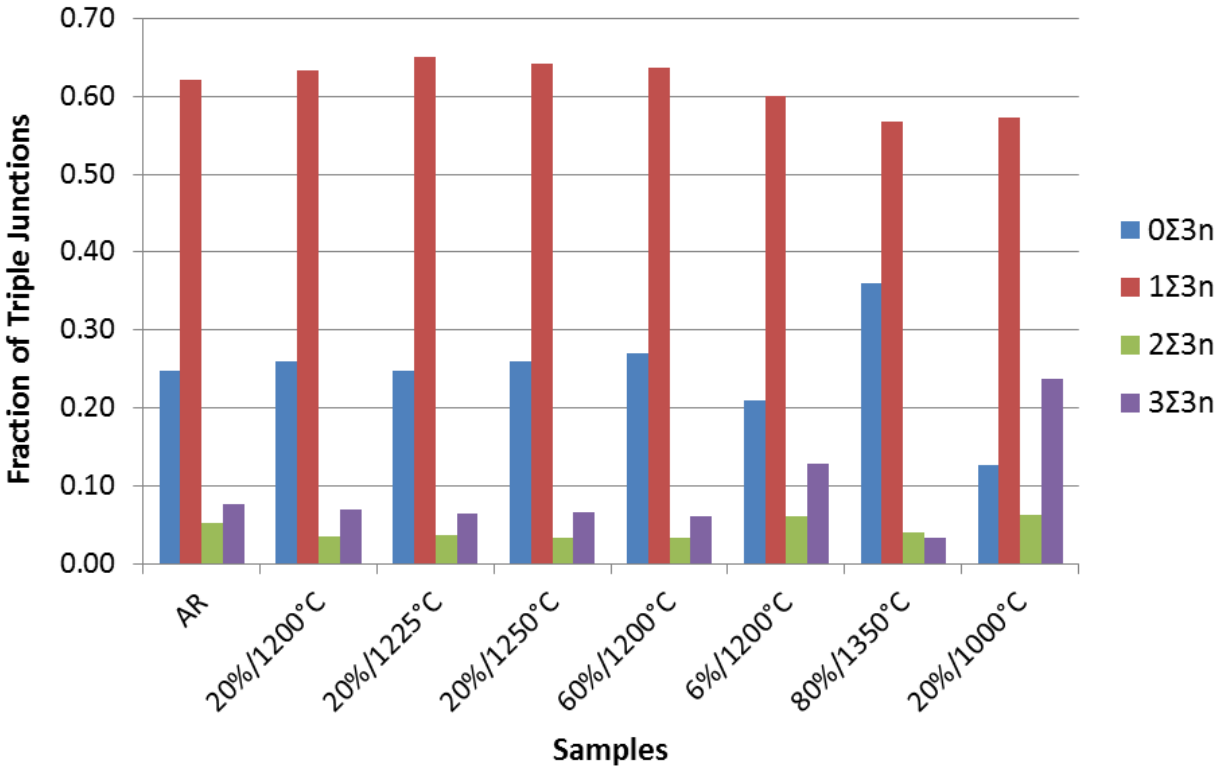


Figure 5.11 Triple junction distributions for selected samples.

The breakup of the RHGB network due to an increase in Σ3 length will influence the creep performance of Alloy 800H [15]. Therefore, the break-up of the RHGB must be minimized so that creep performance due to variations in grain size and twin population can be analysed.

5.4 Summary

The purpose of this chapter was to study the effect of varying strain and annealing temperature on average grain size, grain size distributions, Σ3 boundary length fraction, and grain boundary network topology. The results from this study have major implications for the successful development of microstructures for creep testing.

Variations in average grain size, from 20.9µm to 177.8µm, were achieved by controlling annealing temperature and time. Samples recrystallized after 60% strains achieved a microstructure with smaller average grain size (87.2µm), compared to samples processed with 20% strain (123.8µm), and annealed at the same temperature, 1200°C, and time, 120 minutes.

Samples strained to 60 and 80%RT produced microstructures with tighter grain size distributions compared with samples strained to 20%RT. The ability to control the grain size distribution by varying strain allows for the preparation of samples with varied distributions for creep testing. The processing was not able to produce samples with abnormal grain growth, which in the past have been shown to affect creep performance due to the arrangement of large grains surrounded by small grains.

Results showed that the Σ3 boundary length fraction could be influenced by controlling the growth rate during recrystallization and strain annealing. Using variations in strain and annealing temperature, this study was able to produce samples with Σ3 length fraction fractions between 42.5 and 57.4%. Employing trace analysis found that the coherent Σ3 (twin) length fraction ranged between 25.0% and approximately 32.0%.

While it is possible to increase the length fraction of coherent twin boundaries to 32.0%, the break-up of the RHGB will also have an impact creep performance, making it difficult to differentiate the effect of the network topology with that of grain size and twinning. The results from triple junction analysis suggest that a range of twin boundary length fractions between 25.0 and 28.0% is possible, while at the same time keeping the RHGB network topology consistent.

Chapter 6 Alloy 800H Creep Testing

6.1 Introduction

Chapter 6 details the creep testing performed on Alloy 800H. Creep testing was performed in custom built testing rigs at 980°C and stresses of approximately 13.5MPa. Samples were cut from plate Alloy 800H, processed to produce range of average grain sizes, 61.8-243.7µm, coefficient of variation, 0.764-0.956, and coherent twin length fractions, 24.6% to 31.5%. The effect of average grain size, grain size distribution, and coherent twins on steady-state creep rate and creep ductility is discussed as well as the microstructural changes that occur during creep.

6.2 Creep Rig Design and Testing Methodology

6.2.1 Creep Specimen Design

The Alloy 800H material used for creep testing was received as plate from ThyssenKrupp VDM Australia Pty. Ltd. A scaled-down version of the pin-holed test specimen specified in ASTM E8 [155] was used, with cross-sectional dimensions of 4mm x 3 mm, and a gauge length of 30mm, Figure 6.1 .

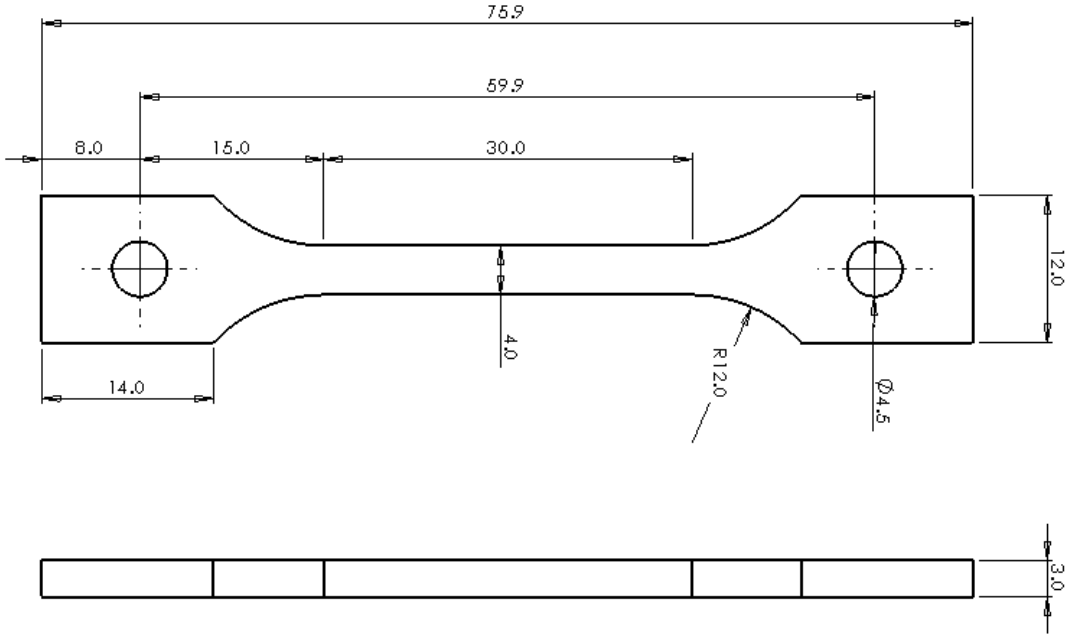


Figure 6.1 Dimensions of the rectangular creep specimens.

6.2.2 Creep Testing Apparatuses

Two creep testing apparatuses were used in assessing the creep performance of Alloy 800H. The first generation (G1) creep apparatus was designed to solely assess the steady-state creep rate. The design of G1 was based on the successful testing apparatus designed for the testing of Alloy 800H at similar conditions [15]. This design was revised to test four samples simultaneously in series, instead of the two samples able to be tested in the G1 apparatus. The second generation (G2) of creep apparatus was for the testing of HP alloy at temperatures and stresses in excess of those proposed for Alloy 800H [156]. The G2 creep apparatus had the additional feature of testing to rupture, thus obtaining the complete creep response.

Table 6.1 summarises the design requirements for the accelerated creep testing apparatuses. In general, the major design elements were consistent across both apparatuses.

Table 6.1 Design Requirements for accelerated creep testing apparatuses.

Testing Temperature	700-1050°C (±1°C)
Load	Up to 50kg
Test Duration	500-4000 hours
Samples per Apparatus	4
Extensometer	Linear variable differential transducer (LVDT)
Data Acquisition	National Instruments Data Acquisition (DAQ)

During service, Alloy 800H pigtails operate at approximately 870°C and stresses between 8-10MPa, for durations of up to 100,000 hours. Obviously, testing for such a prolonged period within a laboratory environment is impractical, and thus the testing temperature and/or stress must be increased in order to obtain failure of the creep samples within an acceptable timeframe. According to Special Metals material data for alloy 800H/HT [2], a stress of 13.5MPa at 980°C would provide a rupture time of approximately 6 weeks. Drabble [15] showed that testing at 13.5MPa and 980°C would ensure that the diffusional creep mechanisms experienced by pigtails at typical service conditions would still dominate at the accelerated test conditions.

Since it was anticipated that these apparatus would be used for future testing, the temperature range was extended to be between 700-1050°C, and the maximum load was extended to 50kg (40MPa for the current specimen design). In addition to the temperature requirement, the maximum allowable temperature variance was limited to ±1°C. The temperature control between individual samples was

possible with the inclusion of a sodium filled isothermal furnace liner (IFL). Ni-based alloys and ceramics were used for the sample grip components and load rods. Additionally, high temperature N-type thermocouples were used to ensure that the temperature measurement and furnace control remained stable throughout the duration of the test.

Figure 6.2 shows the temperature profile with and without the IFL for the furnace used in apparatus G1 . The furnace was orientated vertically with the position of 0mm representing the top of the furnace. The top and bottom furnace openings were packed with insulation similar to the creep test setup. The temperature profiles show the improvement achieved in temperature uniformity by including an IFL with temperature never deviating more than ±1°. A similar result (±1°) was observed in apparatus G2 with the results documented in Ref. [156].

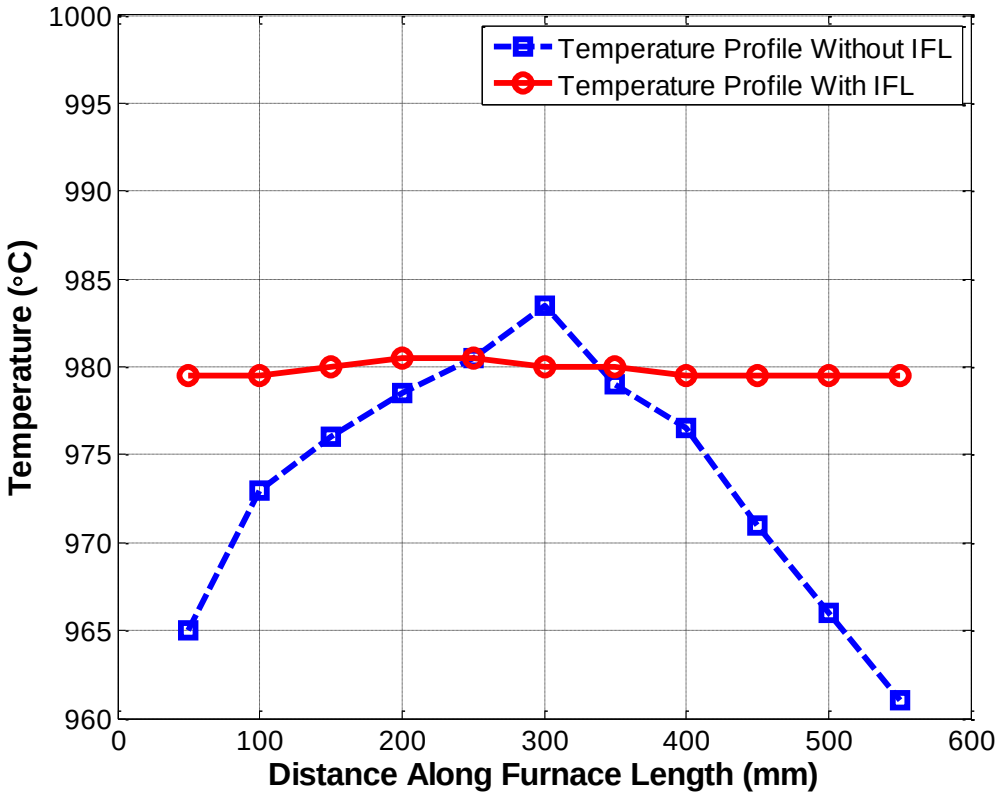


Figure 6.2 Temperature profile along the furnace length with and without the IFL.

Due to the considerable length of time required to perform individual tests, it was decided that the testing apparatus would be designed to simultaneously test multiple creep samples. During detailed design of the apparatuses it was found that the optimal number of samples was four. Any more than four samples would result in excessively large furnace cavities affecting temperature control and risking interference between components.

An extensometer is a device that measures the samples elongation during the creep test. Linear variable differential transducers (LVDTs) were selected for this research due to their relatively low cost (<$1000NZD) and high resolution (≈1µm). The LVDT consists of a magnetic core surrounded by a separate cylindrical sheath which contains AC solenoid coils, Figure 6.3. Linearly displacing the core via a pushrod changes the inductance across the solenoids which vary the output voltage. Hence, by knowing the relationship between the pushrod displacement and the output voltage, accurate linear measurements (e.g. elongation) can be measured.

The measurement of creep strain using LVDTs has shown to be successful in previous accelerated creep testing apparatuses at the University of Canterbury [15, 156, 157]. An extensive review of the advantages of LVDT extensometers was presented by Dr. Takanori Sato [157].

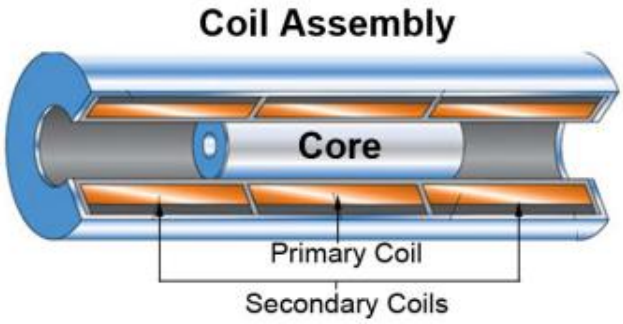


Figure 6.3 Schematic showing the internal mechanism of a linear variable differential transducer (LVDT) [158].

Analogue signals transmitting temperature and LVDT extension data from the test environment were converted to digital values via a National Instruments Data Acquisition (DAQ) system. A project was created in LabVIEW to display the data in real time and record the measurements at a sampling rate of 0.002Hz. The hardware used allowed the collection of data to proceed for multiple months without any issues with program stability.

6.2.3 Extracting Steady-State Creep Rate from Creep Data

The LVDT extension measurements often exhibited periodic variations, typically in the order of ±5µm. The periodicity of these variations occurs over approximately 24 hour periods, are therefore are likely the result of ambient temperature fluctuations between day and night. Figure 4.6 shows temperature fluctuation for measurements performed on the creep rig frames of a period of 700 hours. The measurements were performed in summer for apparatus G1 and in winter for apparatus G2. Both rigs

show periodic fluctuations in frame temperature similar to variations observed in the LVDT data. On closer examination of the data, Figure 6.5, the periodicity of the variations follows the same 24 hour cycle also observed in the LVDT data.

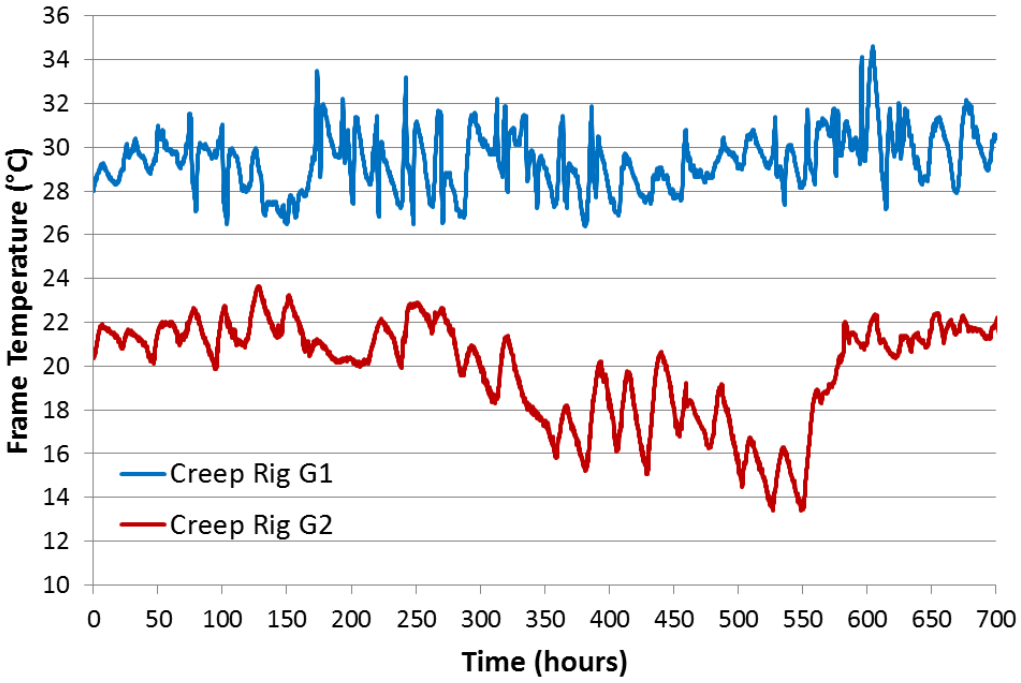


Figure 6.4 Temperature of the creep rig frames of a period of 700 hours. The data collected for Creep Rig G1 was performed in summer and the Creep Rig G2 in winter.

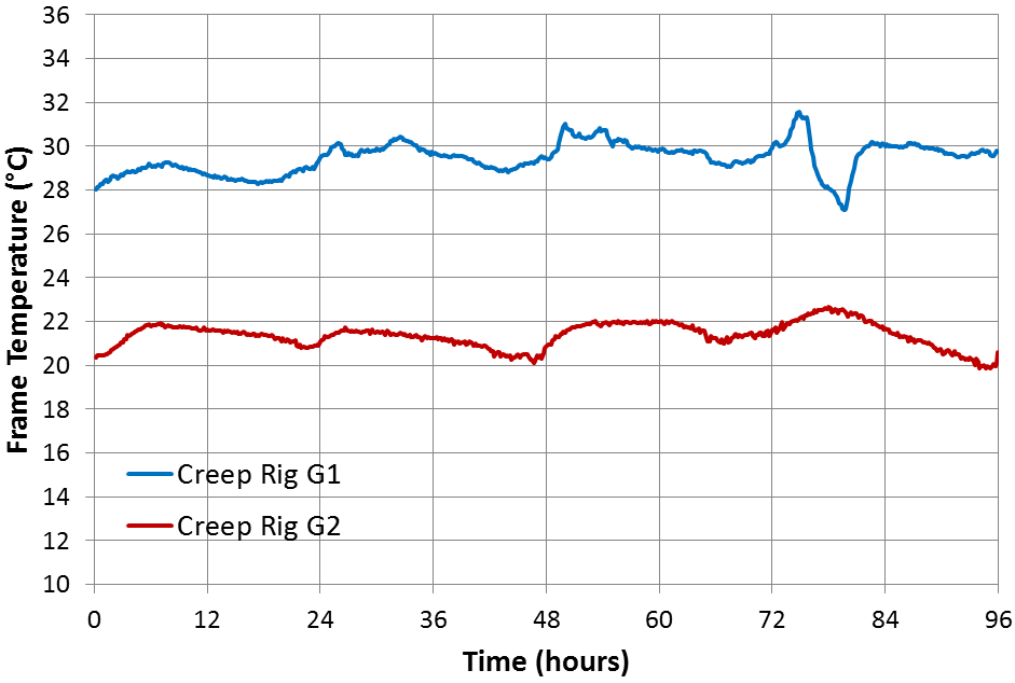


Figure 6.5 Section of data from Figure 6.4 showing the periodic fluctuations of a four day (96 hours) period.

Figure 6.6 is a schematic of the apparatus G2. The grip/sample/LVDT assemblies are fixed to the RHS base via 316 Tensile Load Bars and to the top via the Lever arm assembly. This arrangement would result is any thermal expansion is the rig housing (RHS frame) possibly effecting the grip assembly and LVDT measurement. Calculating the thermal expansion of the RHS uprights on apparatus G2 (length approximately 700mm) for a temperature change of 2°C a change in length of approximately 15µm is possible. This expansion of the rig housing is a possible source of the periodic fluctuations observed in the LVDT measurements.

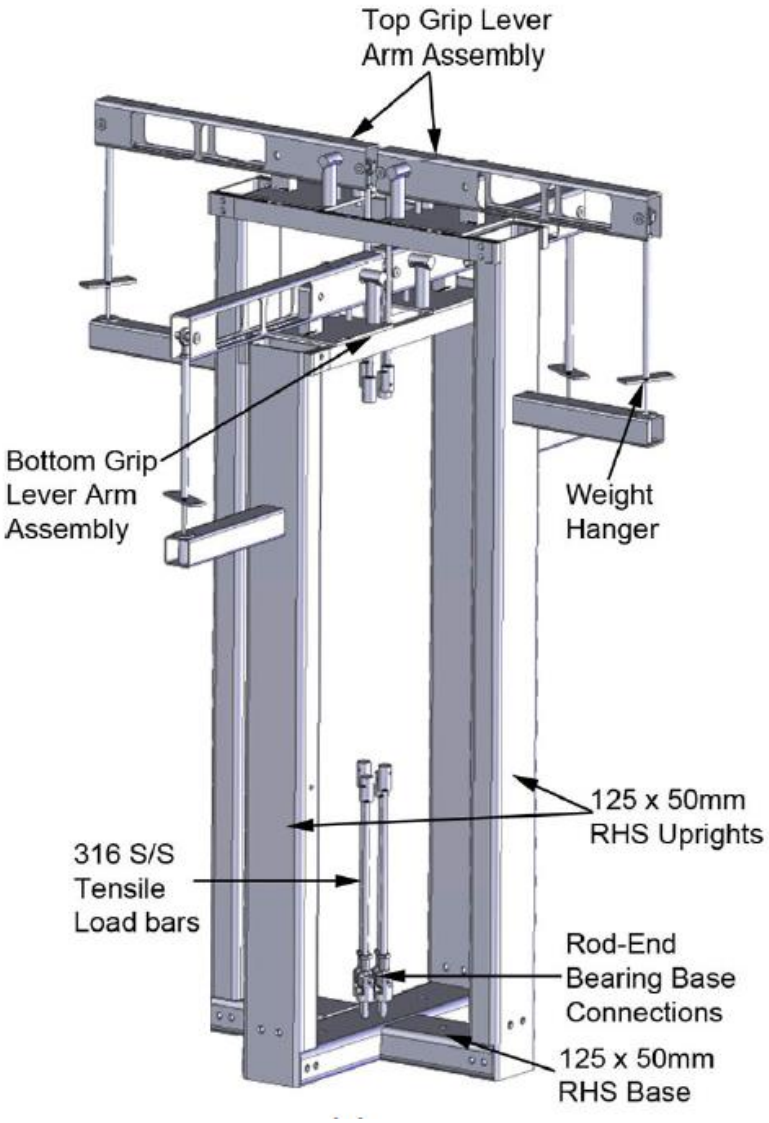


Figure 6.6 Schematic of apparatus G2 showing the sample grip assembly fixed at the bottom and top of the rig housing [156].

The LVDT fluctuations were removed by a data smoothing operation. A moving average operation was applied to the raw data. For each data point, i, a range R_i, was defined according to the Equation 6.1:

$$R_i = \{y_{i-n}\, y_{i-n+1}\, y_{i-n+2} \cdots y_{i+n}\}$$

Equation 6.1

where y_i is the extension value at point i, and n, is the span of the data range. For the present analysis, the span was set at 10 hours either side of the central data point in order to remove the small fluctuations of the extensometer data caused by changes in ambient temperature. For points within 10

hours of the beginning or end of the test, that range was shortened as per Equation 6.2 , where L donates the total number of points:

$$R_i = \begin{Bmatrix} y_1\, y_2 \cdots y_{(2i-1)} \\ y_{(2i-L)}\, y_{(2i-L+1)} \cdots y_L \end{Bmatrix} \quad \begin{matrix} i < n \\ i > (L-n) \end{matrix}$$

Equation 6.2

The smoothed data value at each point, z_i, was then calculated as the mean of the range, $\overline{R_i}$.

The effect of the data smoothing is illustrated in Figure 6.7, which shows a comparison between the raw and smoothed data for an Alloy 800H creep sample.

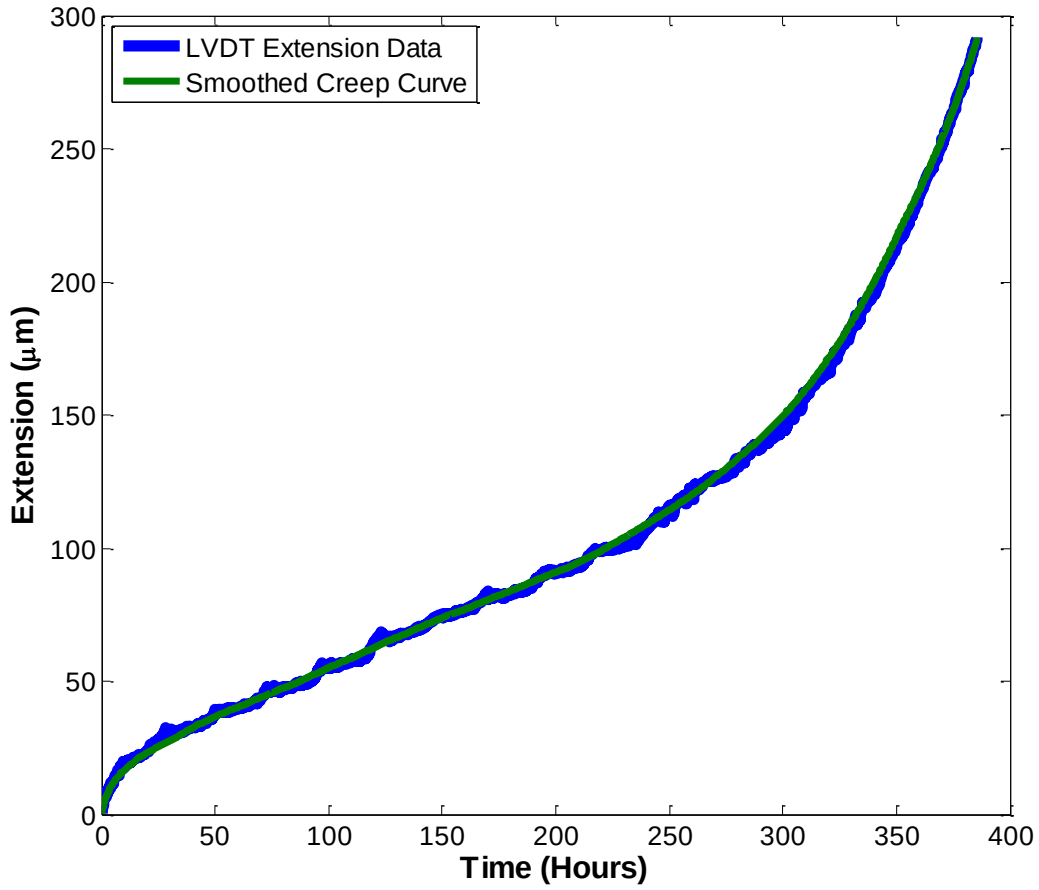


Figure 6.7 Creep curve showing the effect of the moving average data smoothing method.

The instantaneous creep rate was calculated at intervals of 10 minutes, according to a linear least-squares fit with a span of 20 hours, using Equation 6.3:

$$\dot{\varepsilon} = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(y - \bar{y})}$$

Equation 6.3

where the steady state creep rate, $\dot{\varepsilon}$ ($\mu m/hr$), is the slope of the line, the variable represented by x is time (hours, h), and the variable represented by *y* is creep extension (μm). An example creep rate curve is shown in Figure 6.8. Note that the creep rate was not evaluated during the first 10 hours or the last 10 hours of the test, as the data was insufficient to define the span.

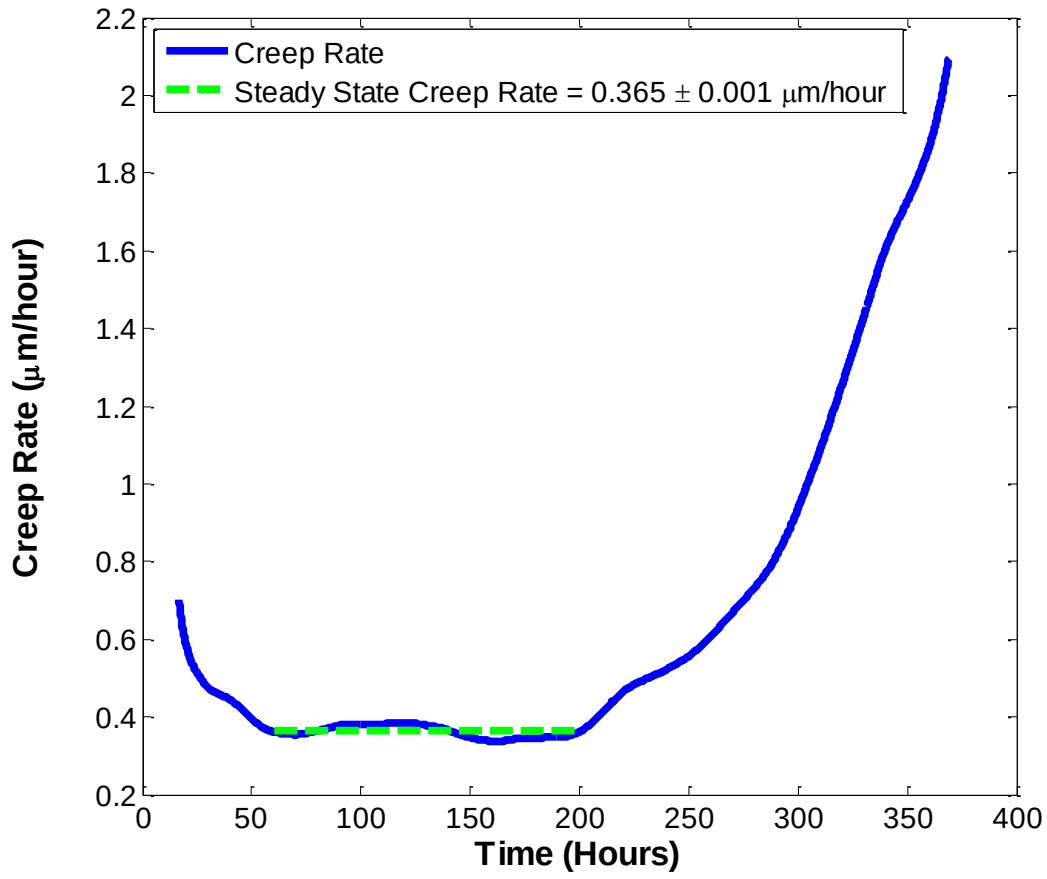


Figure 6.8 Example of creep rate vs time curve

From Figure 6.4, it can be seen that the creep rate remained relatively constant over the period between 60 and 200 hours. Hence, a linear least-squares fit was applied to the smoothed data set for this time period using the Excel function LINEST. The Excel function LINEST also returns an error on the slope of the linear fit, which for the purposes of the current study is used to define the error on the steady state creep rate measurement.

6.2.4 Recording Testing Temperature

Due to the uniform furnace temperature profiles created by the inclusion of the isothermal furnace liners in the creep rig design, only one thermocouple was required to record temperature. Figure 6.9 shows the temperature recorded every 10 minutes over 400 hours. The average temperature over the time period was 980.0°C with a standard deviation of 0.3°C. The maximum temperature recorded during the test was 980.8°C and the minimum temperature was 979.5°C. The average temperature for each test is given in Tables 6.2 and 6.3 along with the standard deviation.

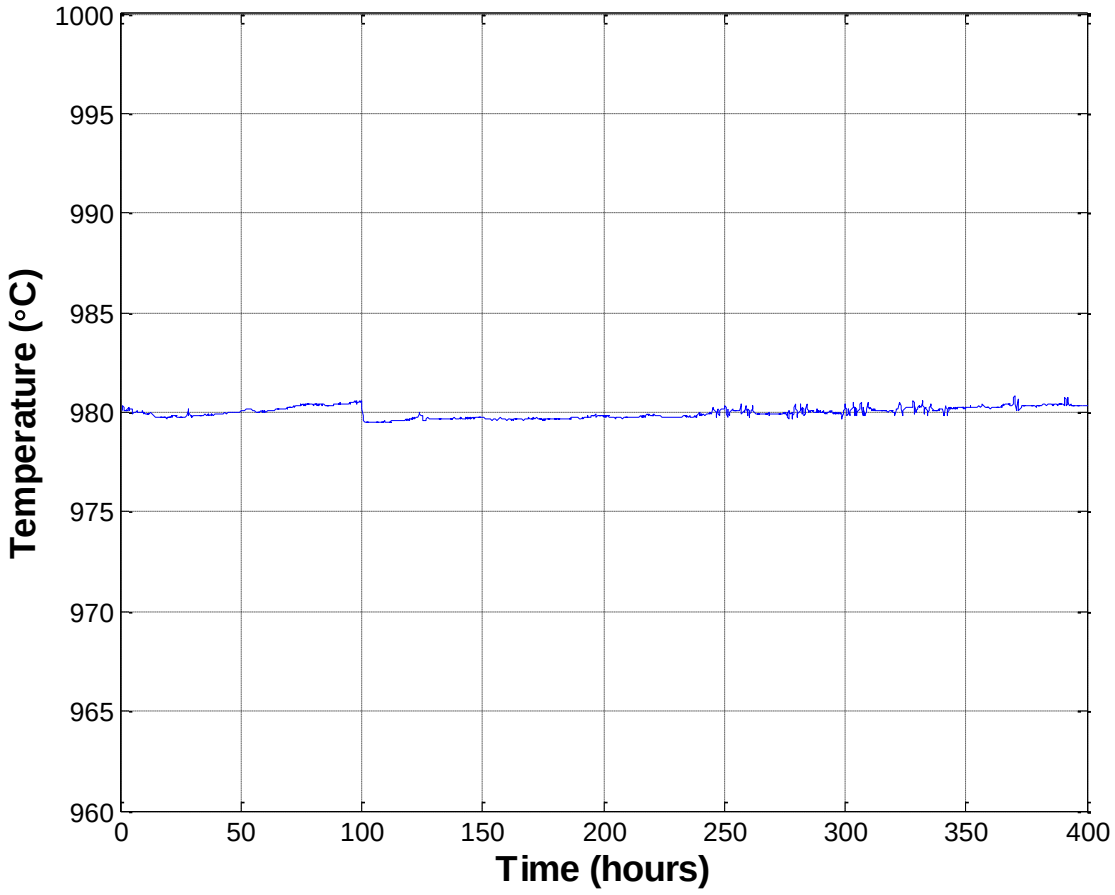


Figure 6.9 Temperature recorded every 10 minutes over a 400 hour period.

6.2.5 Alloy 800H Creep Specimens

Using the results from the Alloy 800H processing study presented in Chapter 5, samples were produced with a range of grain sizes, grain size distributions, and coherent twin fractions. 23 samples were tested,

manufactured, and analysed using the methodologies detailed in Chapter 3. The samples produced had a range of average grain sizes, 55.3-243.7µm, coefficient of variation, 0.764-0.956, and coherent twin length fractions, 24.6-31.5%. 16 samples were successfully creep tested in the G1 apparatus, Table 6.2, and 7 in the G2 apparatus, Table 6.3. One of the rupture creep tests was terminated with one sample not yet failed.

The steady state creep rates, $\dot{\varepsilon}_{ss}$, and creep ductility, $\%\varepsilon_f$, presented in Tables 6.2 and 6.3 were calculated by:

$$\dot{\varepsilon}_{ss}\left[\frac{\%}{hr}\right] = 100\ x\ {}^{\dot{\varepsilon}_{ss}\left[\frac{\mu m}{hr}\right]}\Big/_{Specimen\ Gauge\ Length[\mu m]}$$

Equation 6.4

and:

$$\%\varepsilon_f = 100\ x\ {}^{\varepsilon_f[\mu m]}\Big/_{Specimen\ Gauge\ Length[\mu m]}$$

Equation 6.5

where the specimen gauge length equals 30mm.

Table 6.2 Creep samples tested in steady-state rig (G1). The *Average Grain Size* is given by Equation 3.19, the G*rain Size %Error* is given by Equation 3.20, and the *Coefficient of Variation* describing the width of the grain size distributed grain size measurements is given by Equation 3.23.

Processing			**Characterization**					**Testing**		**Results**
Strain (% RT)	*Temperature (°C)*	*Time (Minutes)*	*Grain Size (µm)*	*Grain Size %Error (±%)*	*Coefficient of Variation (CV)*	*Coherent Twin Length Fraction (%)*	*Σ3 Length Fraction (%)*	*Creep Stress (MPa)*	*Temperature (°C)*	*Steady-State Creep Rate (%/hr)*
20	1275	10	92.7	10.3	0.904	26.8	51.1	13.2	980.1 ± 0.3	0.00114
20	1275	20	119.2	14.1	0.909	26.4	49.9	13.3	980.1 ± 0.3	0.00068
20	1275	40	158.9	12.1	0.940	26.7	49.7	13.4	980.1 ± 0.3	0.00038
20	1275	60	174.8	13.2	0.914	29.3	54.0	13.7	980.3 ± 0.5	0.00019
20	1275	90	211.6	10.9	0.956	27.5	53.0	13.6	980.0 ± 0.3	0.00023
20	1275	180	243.7	13.4	0.942	31.5	55.9	13.3	980.1 ± 0.3	0.00023
20	1150	10	55.3	12.6	0.884	27.2	50.4	13.3	980.3 ± 0.5	0.00217
20	1150	20	64.8	10.1	0.925	26.0	54.1	13.7	980.0 ± 0.3	0.00154
20	1150	30	71.8	11.2	0.945	27.6	51.3	14.0	980.0 ± 0.3	0.00122
20	1150	40	75.7	12.2	0.948	28.4	52.2	13.9	980.0 ± 0.3	0.00108
20	1150	50	81.8	11.1	0.889	28.2	52.7	13.9	980.0 ± 0.3	0.00088
60	1200	20	85.3	11.6	0.948	25.4	47.8	13.4	980.4 ± 0.4	0.00116
60	1200	40	93.2	12.3	0.887	25.1	47.2	13.8	980.3 ± 0.5	0.00056
60	1200	60	94.1	10.0	0.852	26.2	47.5	13.7	980.4 ± 0.4	0.00099
60	1200	120	109.3	14.2	0.800	30.2	51.5	13.8	980.4 ± 0.4	0.00043
60	1300	30	160.1	13.2	0.804	25.1	47.5	13.5	980.0 ± 0.3	0.00019

Table 6.3 Creep samples tested in rupture rig (G2). The *Average Grain Size* is given by Equation 3.19, the G*rain Size %Error* is given by Equation 3.20, and the *Coefficient of Variation* describing the width of the grain size distributed grain size measurements is given by Equation 3.23.

Processing			**Characterization**					**Testing**			**Results**	
Strain (% RT)	*Temperature (°C)*	*Time (Minutes)*	*Grain Size (μm)*	*Grain Size Error (±%)*	*Coefficient of Variation (CV)*	*Coherent Twin Length Fraction (%)*	*Σ3 Length Fraction (%)*	*Creep Stress (MPa)*	*Temperature (°C)*	*Failed?*	*Steady-State Creep Rate (%/hr)*	*Creep Ductility (%)*
20	1225	10	78.5	12.1	0.939	24.6	49.0	13.6	980.7 ± 0.1	✓	0.00141	39.5
20	1225	30	125.7	11.1	0.945	29.2	55.2	13.8	980.7 ± 0.1	✓	0.00062	13.8
20	1225	60	145.8	10.8	0.893	26.0	48.9	13.8	979.7 ± 0.4	✓	0.00054	19.6
60	1200	90	97.0	11.4	0.878	25.8	47.8	13.7	980.7 ± 0.1	✓	0.00048	35.1
60	1200	150	119.8	12.6	0.890	25.2	47.1	13.7	980.7 ± 0.1	✓	0.00040	25.7
60	1300	60	199.8	11.3	0.764	26.0	46.1	13.6	979.7 ± 0.4	✖	0.00019	16.4
60	1300	120	232.6	12.2	0.765	26.2	46.9	13.7	979.7 ± 0.4	✓	0.00011	21.3

6.3 Results and Discussion

6.3.1 Sample Characterization

Figure 6.10 shows the average grain sizes for the creep specimens grouped by their processing conditions (RT/Temp). The samples annealed at 1275°C and 1300°C produced the largest average grains size, 243.7µm and 232.6µm, respectively. The smallest grain size was 55.3µm for the sample prepared by cold rolling to 20% RT and annealing at 1150°C for 10 minutes. The average grain sizes compare closely to the average grain size measurements performed in Chapter 5. For example the 20%/1225°C/60min sample discussed in Chapter 5 had an average grain size of 125.2µm, within 13% of the value shown in Figure 6.10 prepared using the same processing conditions. The samples prepared for creep testing covered the range of ASTM grain sizes 5.5 to 1.5.

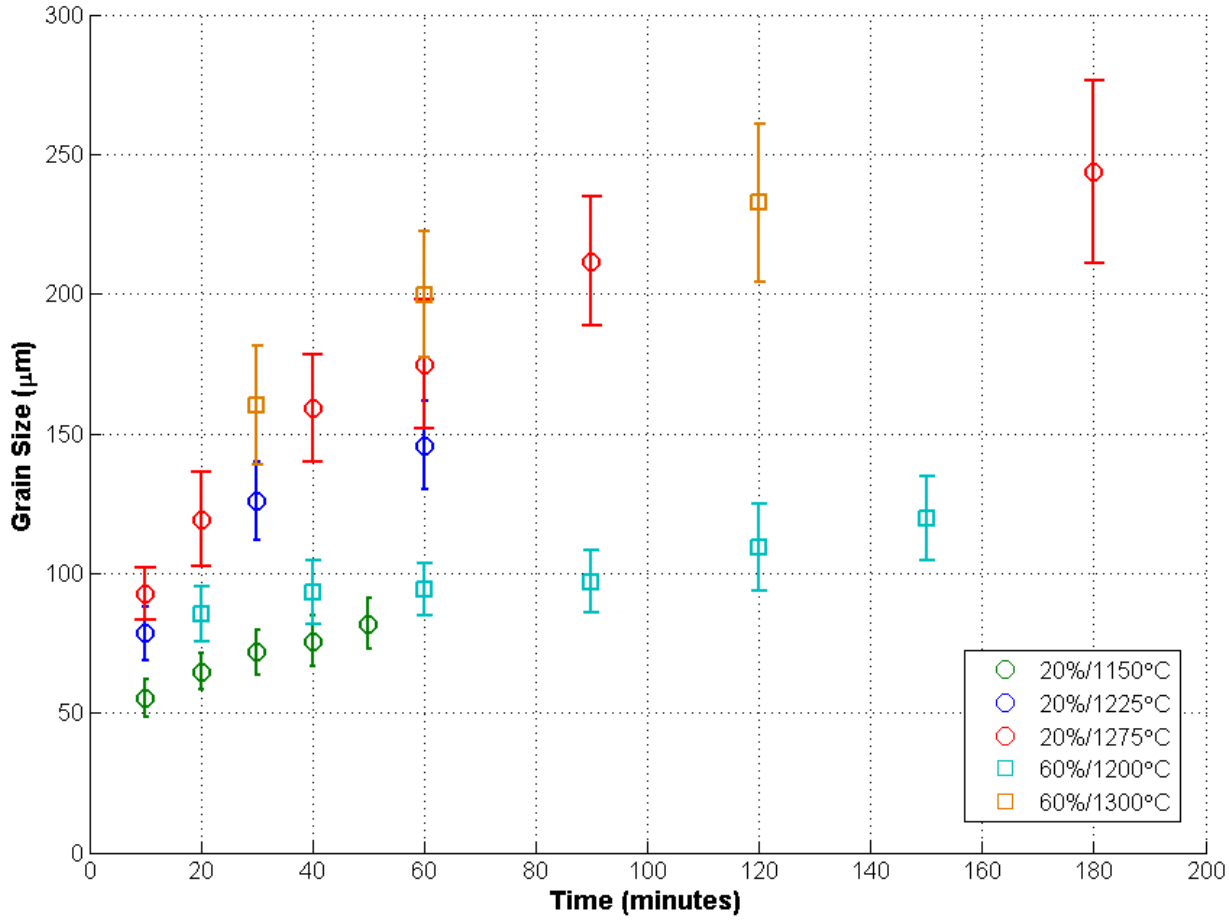


Figure 6.10 Average grain size for samples strained to 20% RT and annealed at 1150, 1225, and 1275°C.

Figure 6.11 shows the average distribution of boundary types and average coefficient of variation for the five sample groupings. The coefficient of variation was lowest for samples prepared by cold-working to 60% reduction in thickness, with the tightest distributions belonging to the samples annealed at 1300°C. The fraction of coherent twins (blue) was consistent across all sample groupings with the smallest fraction, 25.8% belonging to the group with 60% strain annealed at 1300°C and the largest fraction, 28.0%, belonging to the sample group with 20% strain and annealed at 1275°C. The total Σ3 boundary length fraction (coherent, blue, plus non-coherent, red) varied between 46.8% and 52.3% for the sample groupings, 60%/1300°C and 20%/1275°C. Overall, the differences in boundary fractions were minimal, and were unlikely to have a measurable effect on the creep performance between the Alloy 800H specimens.

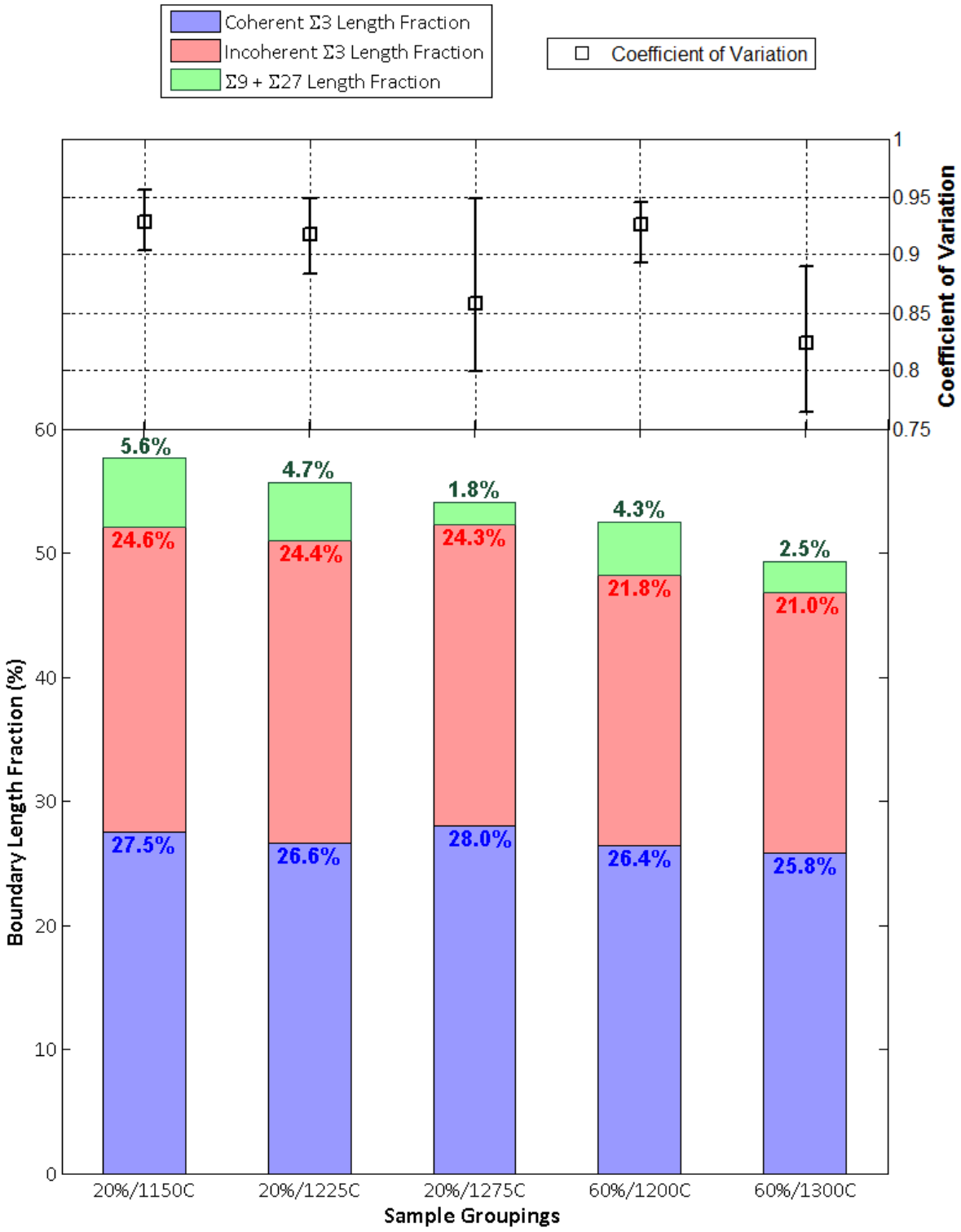


Figure 6.11 TOP: Average coefficient of variation for the grain size distributions for the sample groupings. BOTTOM: Boundary length fractions for $\Sigma 3^n$ (n=1-3) boundary types.

6.3.2 Creep Curves

Figures 6.1 to 6.18 show the creep response for the seven samples tested to rupture in rig G2 compared to samples of similar average grain size tested in rig G1. Generally there was minimal (approximately 10%) difference between steady state creep rates measured from samples of similar average grain size in the two testing rigs. The LVDT data from the sample with the largest grain size, 232.6µm (Figure 6.18), showed fluctuations greater than ±6µm in extension measurement. These fluctuations made it difficult to identify a steady state region and calculate creep rate. Noise measuring approximately ±5µm was previously noted as being typically of the LVDT setup and testing environment. Suggestions for improving creep rigs and testing environment are discussed in future work, Chapter 8.

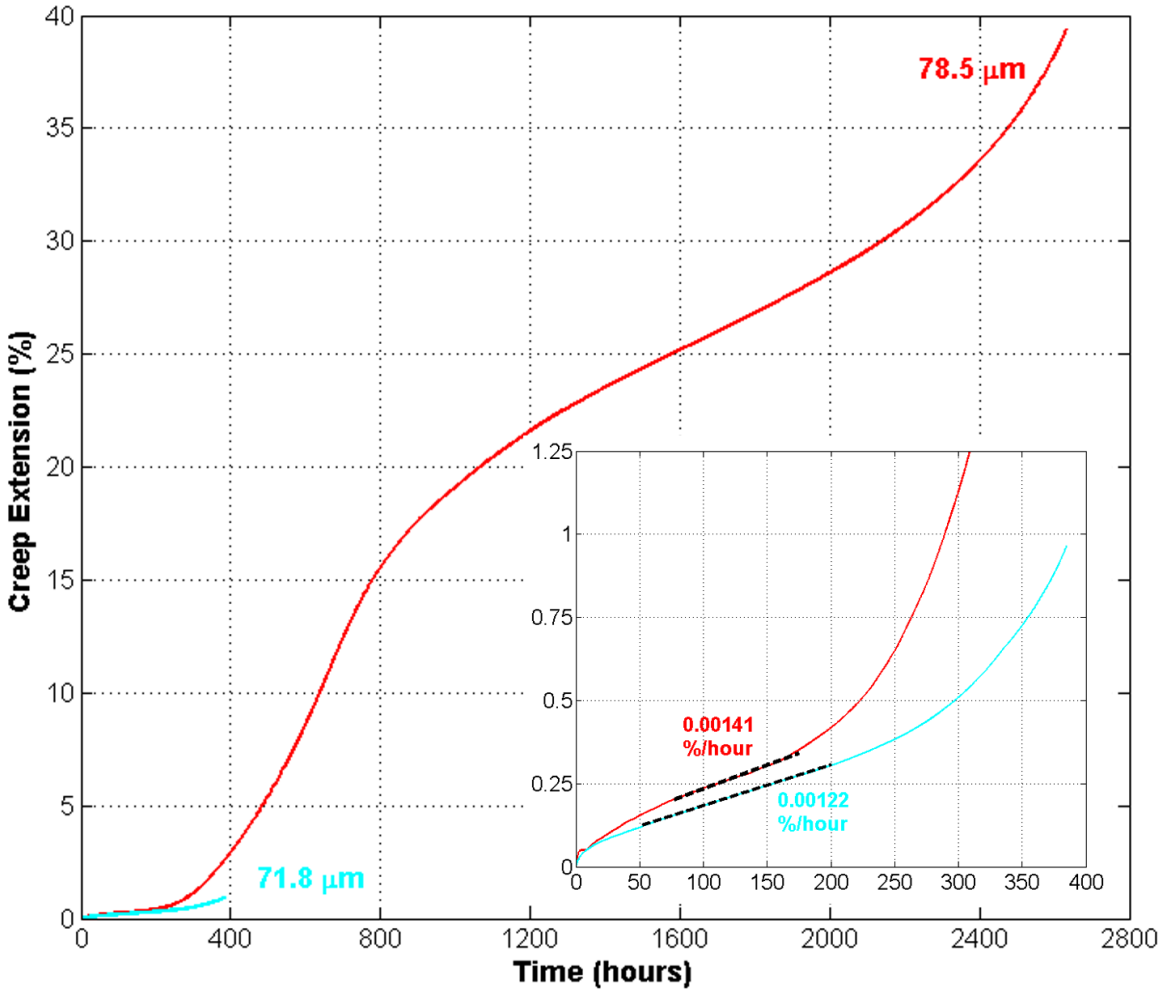


Figure 6.12 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1150°C/30min) and G2 (red, sample 20%/1225°C/10min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

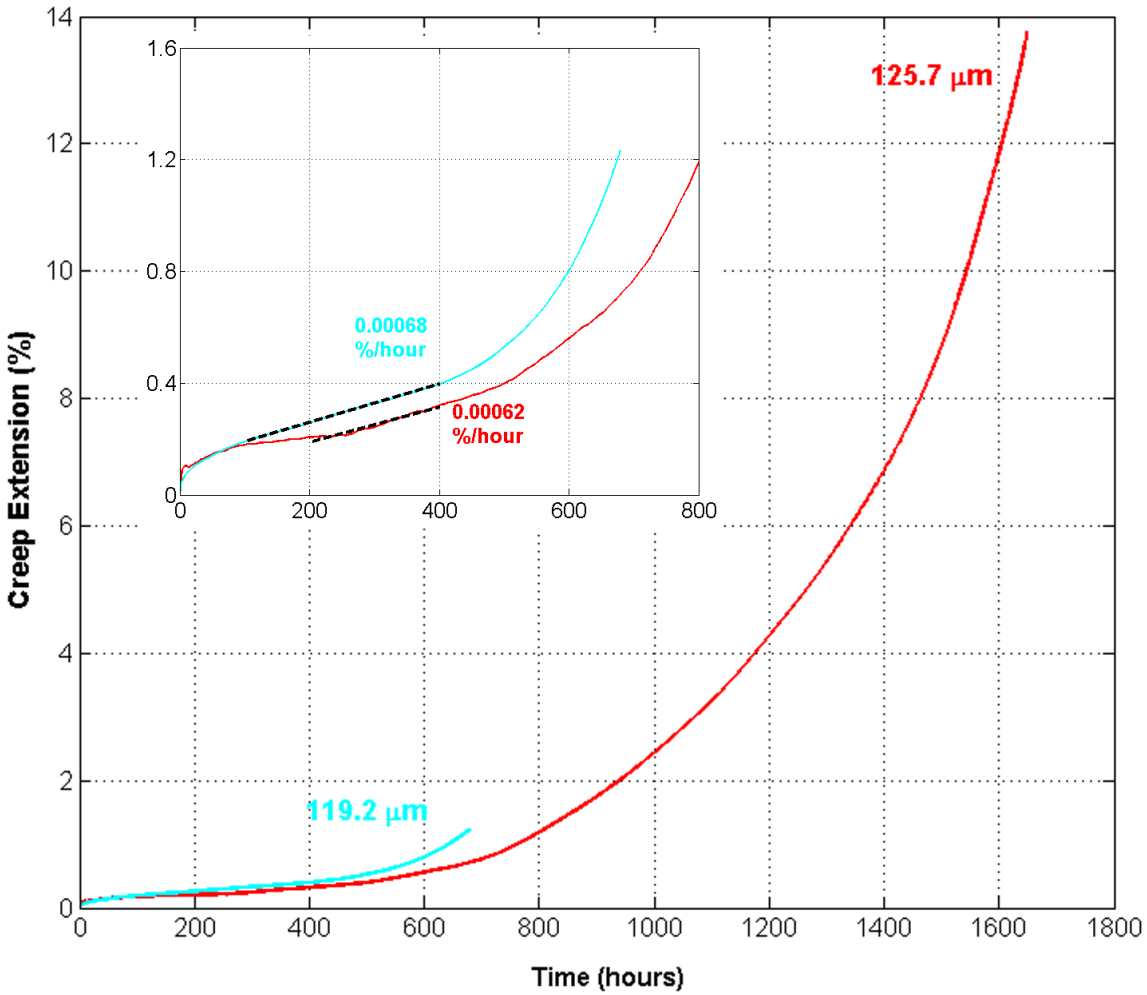


Figure 6.13 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1275°C/20min) and G2 (red, sample 20%/1275°C/20min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

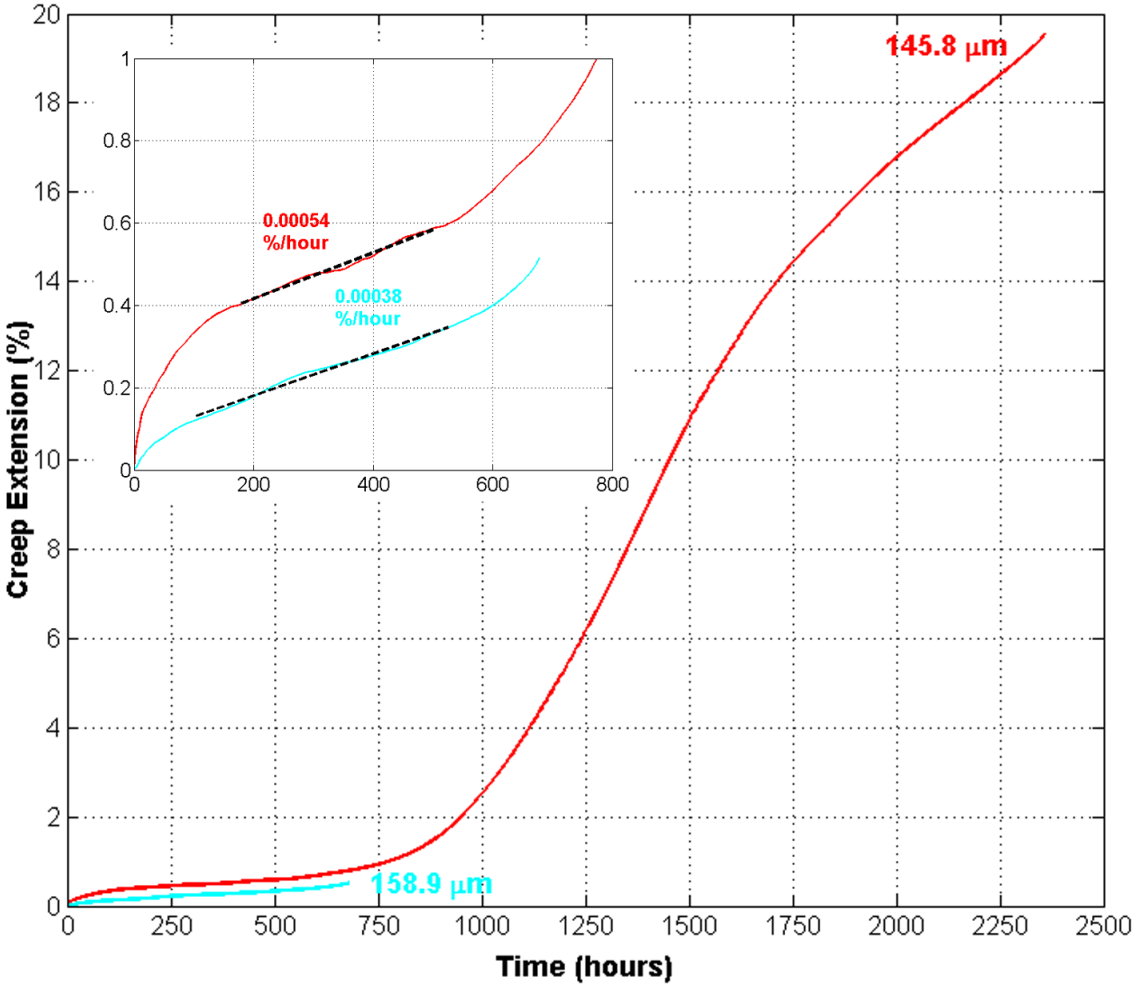


Figure 6.14 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1150°C/30min) and G2 (red, sample 20%/1225°C/60min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

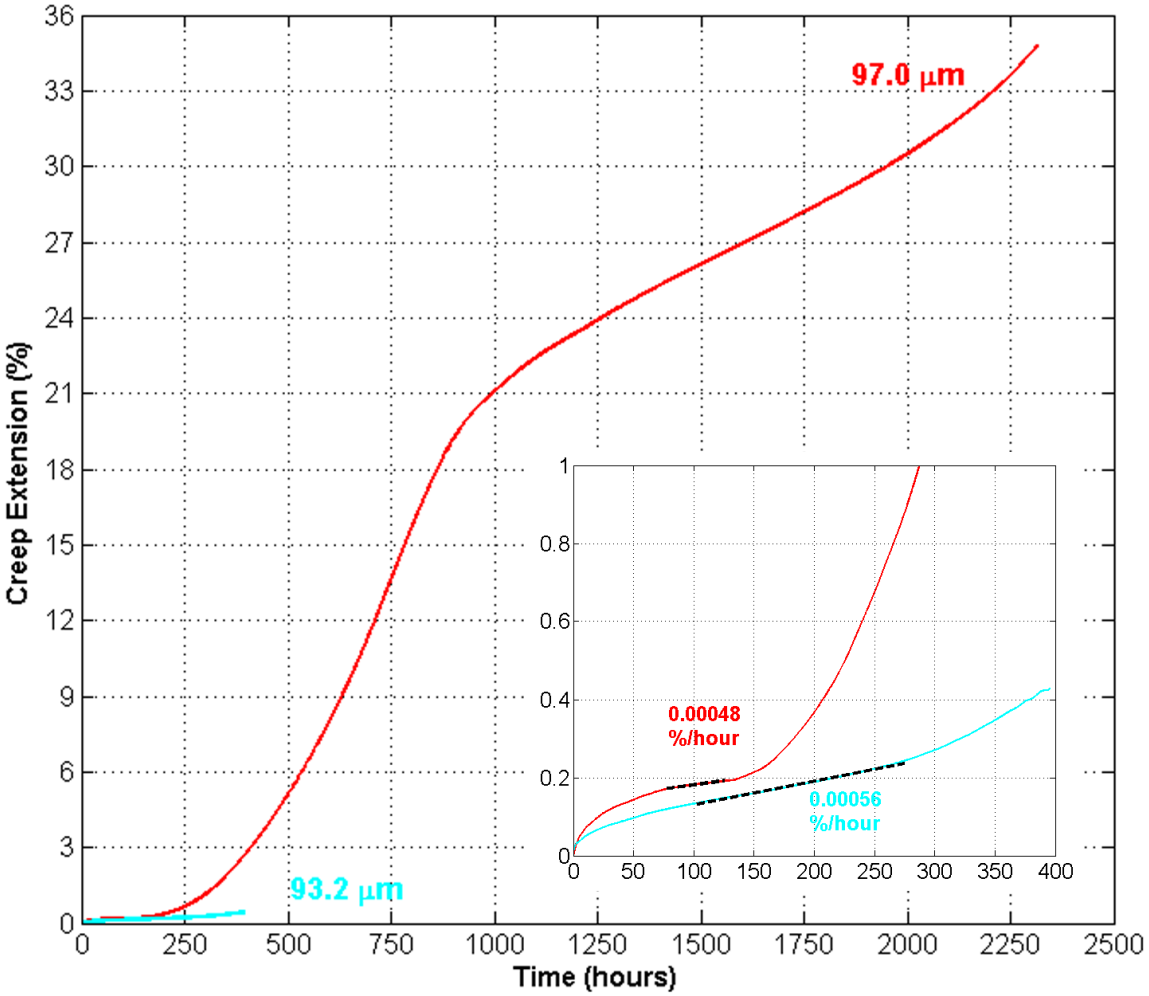


Figure 6.15 Curves comparing the creep responses from samples tested in G1 (blue, sample 60%/1200°C/40min) and G2 (red, sample 60%/1200°C/90min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

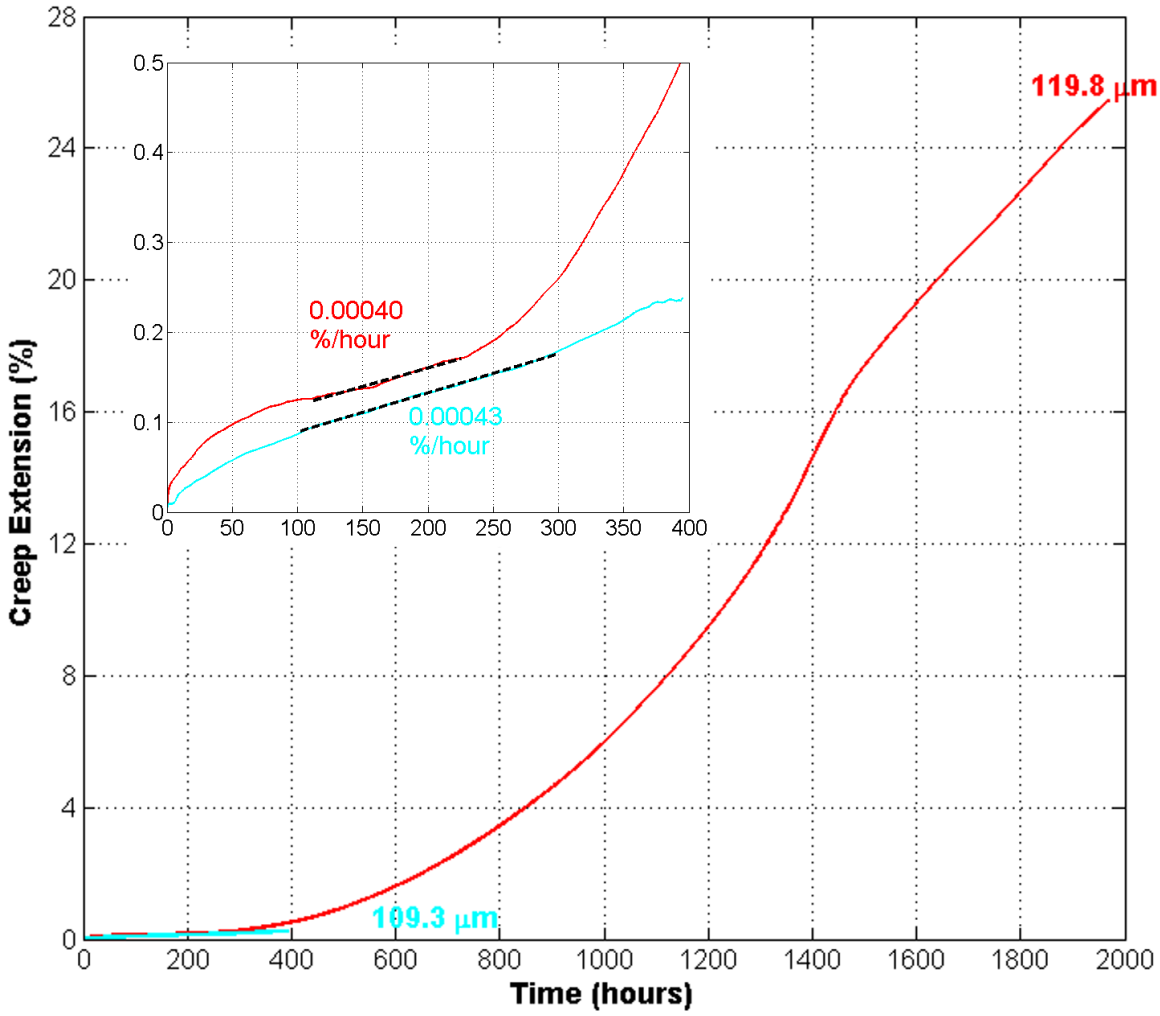


Figure 6.16 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1275°C/40min) and G2 (red, sample 60%/1200°C/150min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

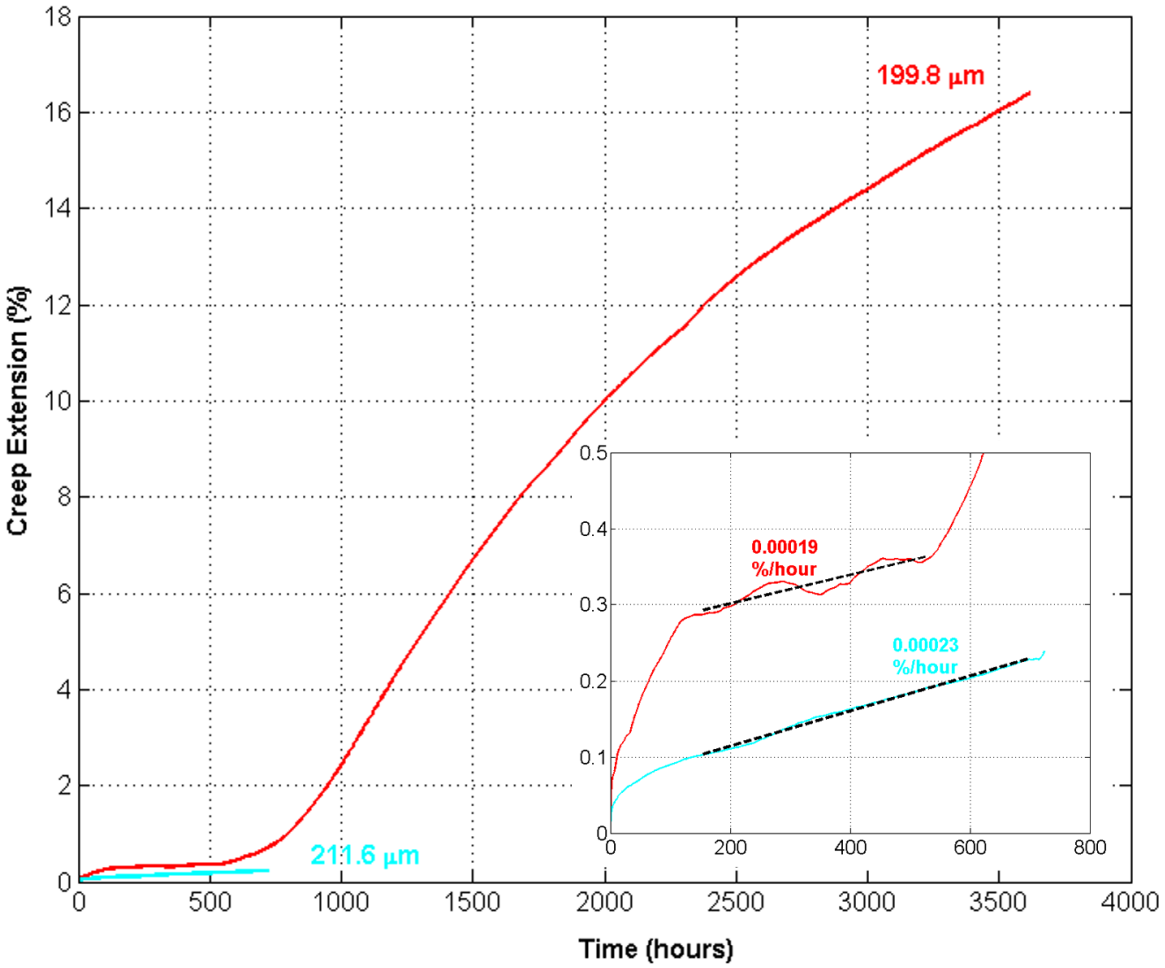


Figure 6.17 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1275°C/90min) and G2 (red, sample 60%/1300°C/60min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

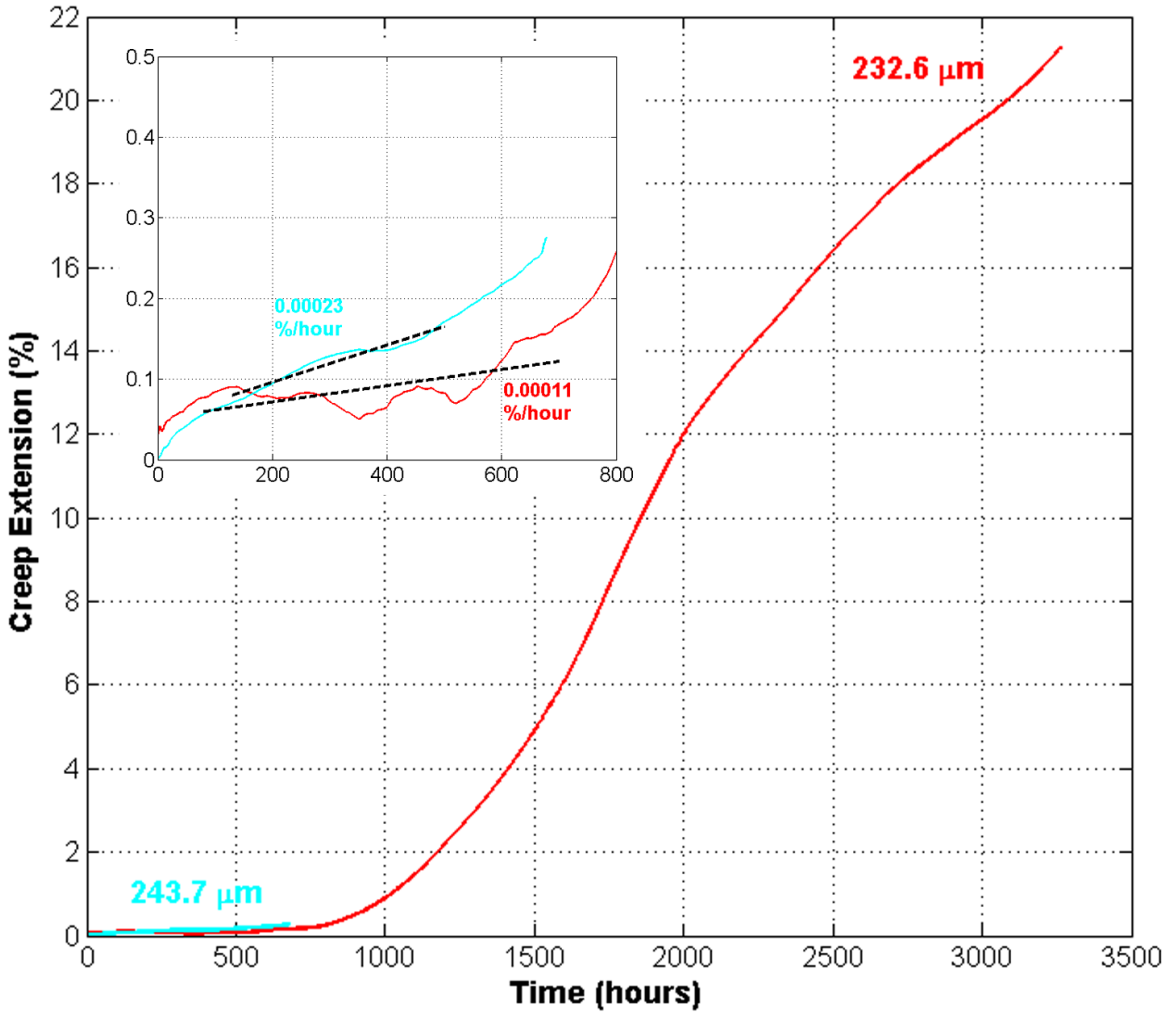


Figure 6.18 Curves comparing the creep responses from samples tested in G1 (blue, sample 20%/1275°C/180min) and G2 (red, sample 60%/1300°C/120min). Insert provides a magnified view of the steady state regions, with creep rate given in %/hour.

6.3.3 Steady State Creep

Figure 6.19 shows the time elapsed and creep strain accumulated with respect to average grain size at the conclusion of the steady state creep regime. The results show that samples with larger average grain sizes have longer steady state creep regimes. For example, the sample with an average grain size of 64.8µm commenced tertiary creep after approximately 125 hours, while it took 700 hours before the sample with an average grain size of 232.6µm was in tertiary creep. Figure 6.19 also shows a minimal amount of creep strain accumulates before the start of tertiary creep. Less than 0.6% extension prior to the start of tertiary creep represents less than 2% of the total accumulated strain at rupture.

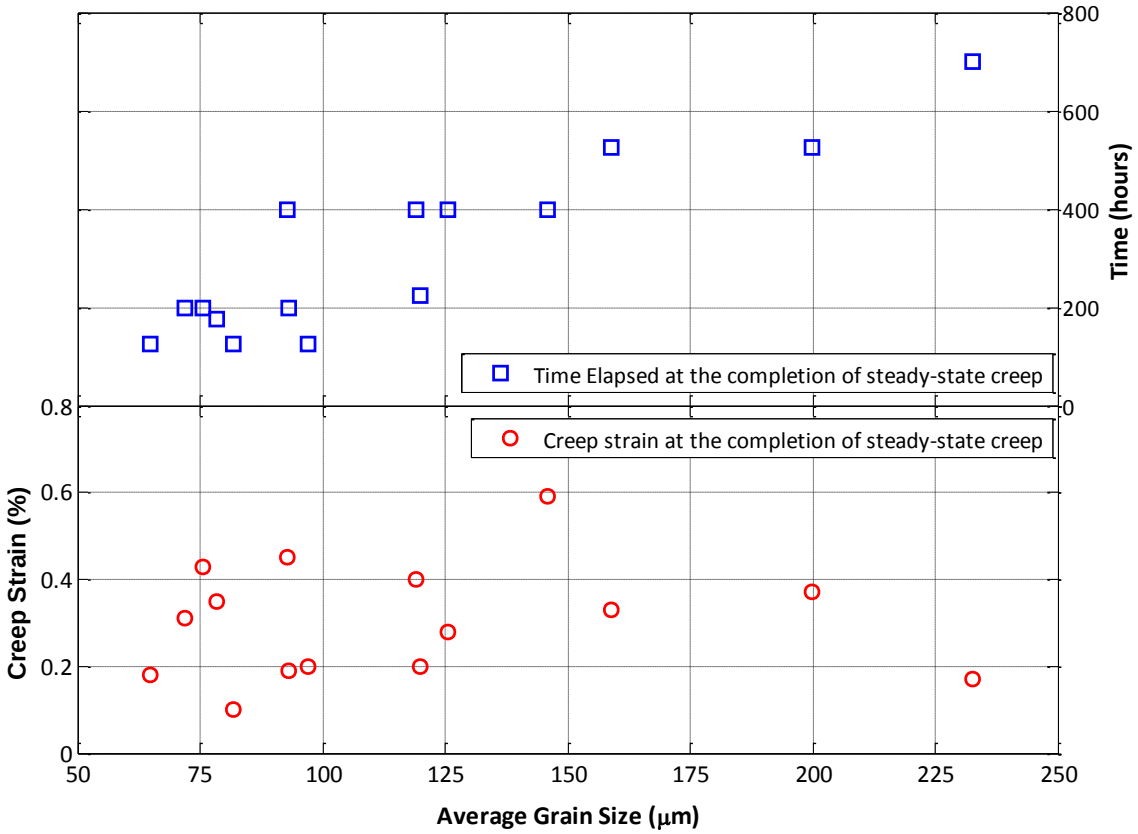


Figure 6.19 The time elapsed (hours) and the creep strain (%) incurred at the completion of steady state creep.

Figure 6.20 shows the steady state creep rate versus average grain size for all creep samples. The steady state creep rates for the individual samples were presented in Tables 6.2 and 6.3. Indicated through a power-law curve fit, a grain size dependence of $\frac{1}{\bar{d}^{1.78}}$ is calculated. The grain size dependence indicates that for the testing conditions, diffusional creep processes dominate. No direct evidence was gathered

to confirm that the dominant creep mechanism was diffusional flow, and additional mechanisms, such as grain boundary sliding, may be grain size sensitive. Additional experiments have been proposed to more conclusively identify the dominant creep mechanism.

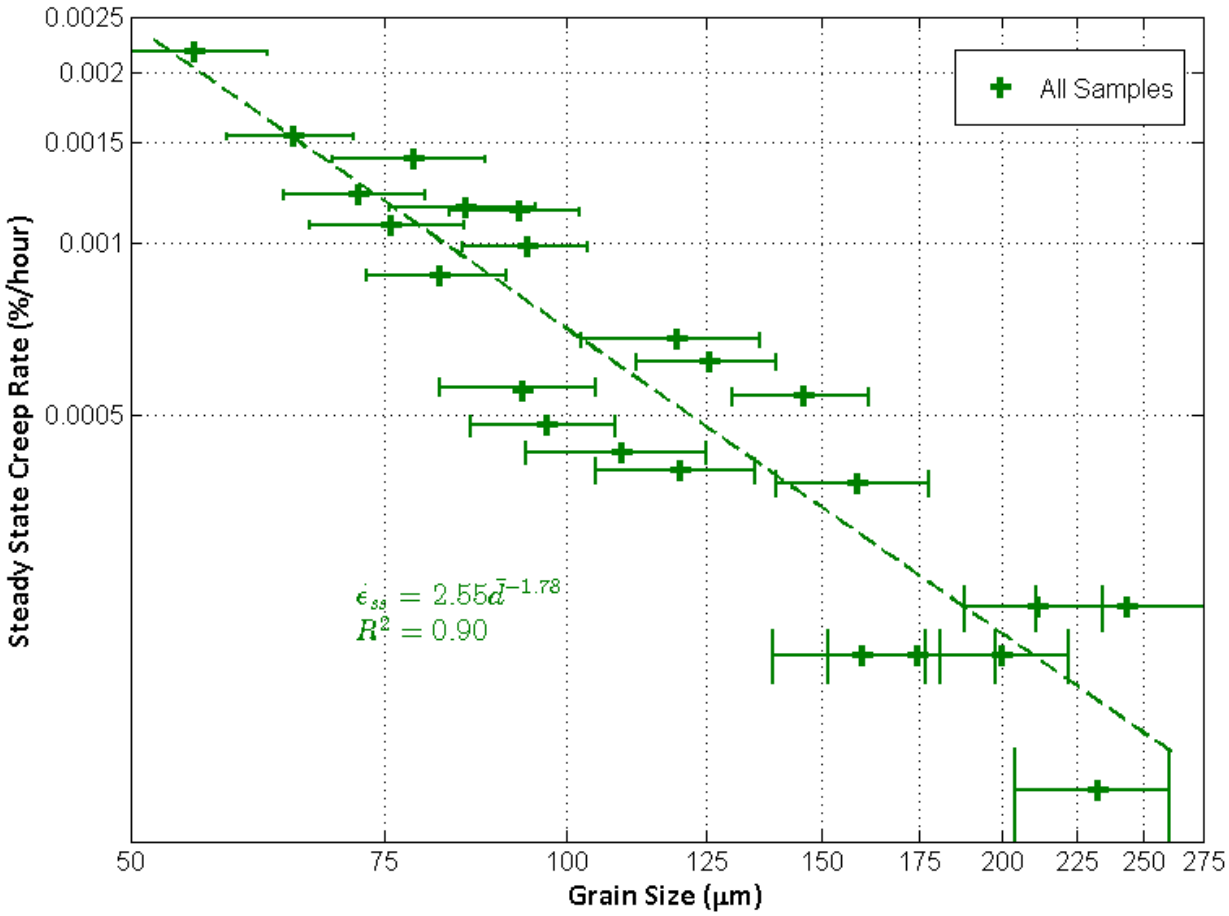


Figure 6.20 Relationship between average grain size and steady state creep rate for all creep samples. Green curve is a power law fit to the creep data.

Figure 6.21 shows the relationship between steady state creep rate and grain size for two sample groupings categorised by the degree of cold-work performed during processing, 20% or 60%. The samples cold-worked to 60%RT typically produced tighter grain size distributions in comparison to the samples cold-worked to 20%RT, coefficient of variation ≈ 0.843 compared to ≈ 0.924, respectively. The results indicate that samples with tighter grain size distributions display slower steady state creep rates. The result is understandable, with the smaller grains at the lower tail of the distribution accumulating creep strain at a faster rate resulting in higher steady state creep rates. Figure 6.22 compares two grain size distributions for samples produced with 20% and 60% cold work and annealed to produce

comparable grain sizes (158.9µm and 160.1µm, respectively). The figure shows that the two samples had similar upper tails, while the sample prepared with 20% cold work had a longer lower tail.

Overall, there was minimal difference (less than a factor of 2) in steady state creep rates between the sample groupings. Coupled with the observation that steady state creep accounts for at most 0.6% of total strain, the effect of steady state creep rate on over all creep life is inconsequential.

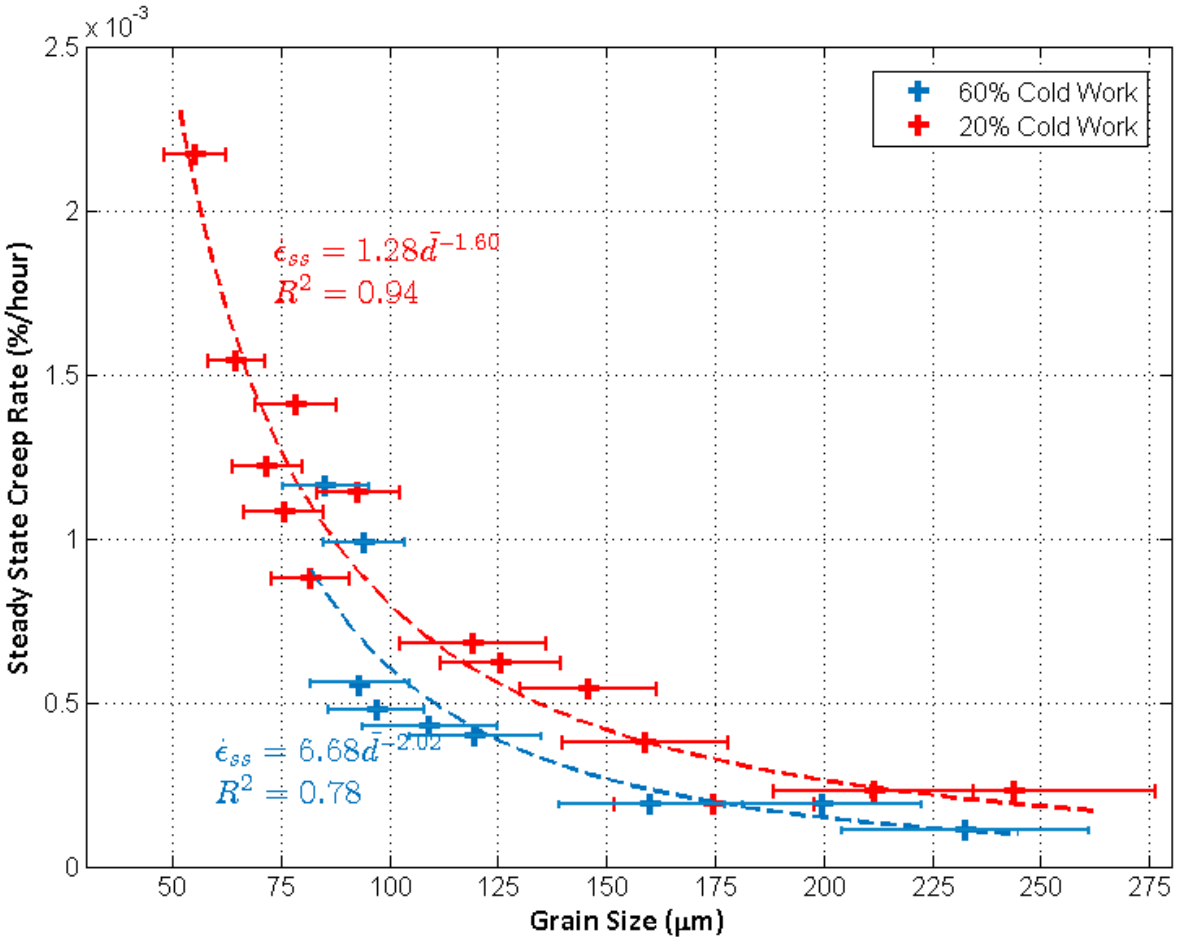


Figure 6.21 Relationship between average grain size and steady state creep rate for all creep samples separated into two groupings representing the difference in cold-work during sample processing.

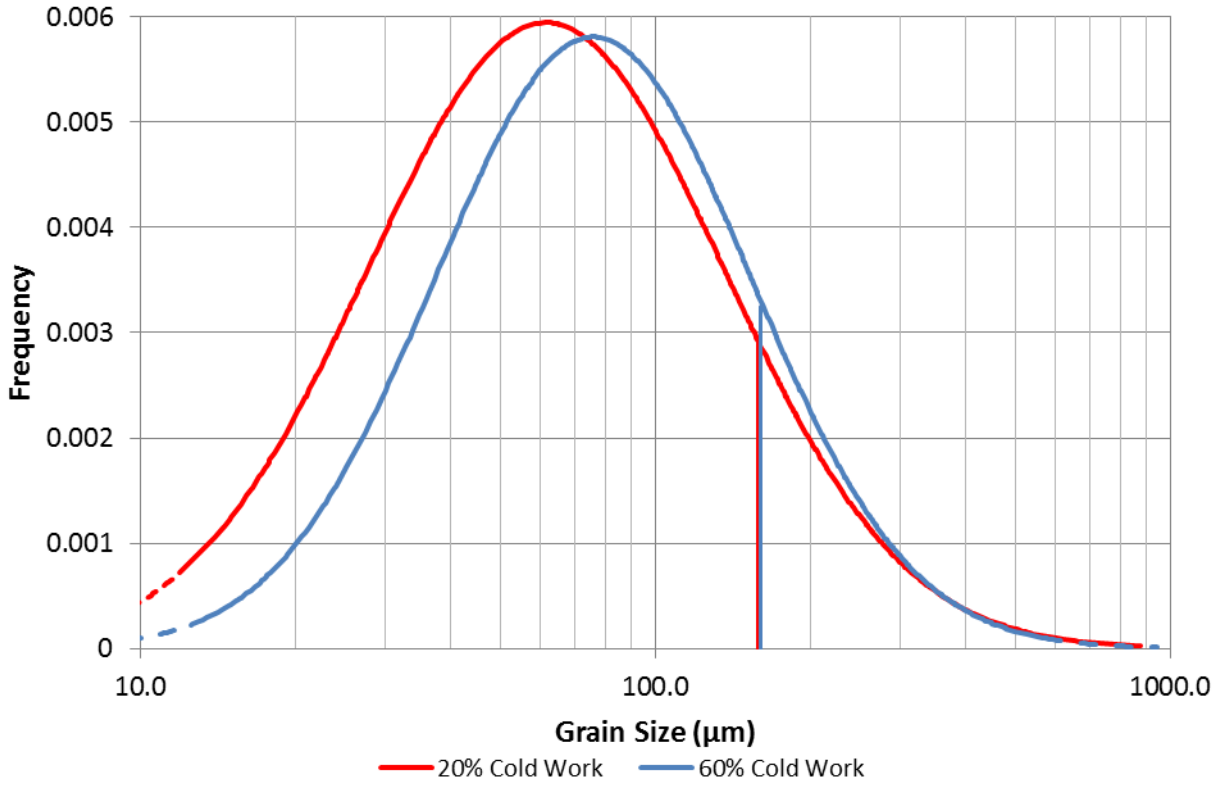


Figure 6.22 Grain size distributions comparing the size of the lower tail for two creep samples produced using 20% cold work and 60% cold work.

Figure 6.23 shows the creep data compared to the predicted creep rates of the Frost and Ashby model [11], derived in Appendix A. The red curves indicate the predicted creep rate for D = 0 and D = 0.68, where D is the standard deviation of the log normal distribution. D = 0.68 is the average standard deviation for all creep samples. While incorporating the grain size distribution into the creep model produced a closer approximation of the experimentally determined steady state creep rate for Alloy 800H, there remains a disparity.

This disparity can be explained in relation to the grain size measurement strategy employed in the current study. §3.6.1 showed a difference of approximately 30% (ASTM grain size number 4.1 compared to 3.3) between the average grain size measured from an optical micrograph, and that measured from an EBSD boundary map. With optical microscopy, all boundaries with straight line morphologies are assumed to be coherent twins and as instructed by ASTM E112 omitted from grain size measurement. However, EBSD mapping combined with trace analysis suggests that approximately 30-50% of the boundaries assumed to be coherent twins may in fact be <110> asymmetric tilt boundaries displaying identical straight line morphologies.

Σ3 boundaries not identified as coherent were included in grain size measurement. A study of an austenitic Ni-16Cr-9Fe alloy [159] concluded that, on average, the low-Σ (Σ<29) grain boundary diffusion coefficient was approximately 12x lower than that of other HABs. Minkwitz et al [154] also showed that the <110> asymmetric tilts positioned approximately 20° away from the {111} (coherent) plane can have diffusivities 10 to 100 times slower than a RHGB. This range of diffusivities is closer to RHGB diffusivities (D_b, Equation 8.9) than lattice diffusivities (D_v, Equation 8.7). Therefore, the decision was made to include them in the measurement of average grain size. Typically, total boundary length included in grain size measurement contained 25-30% non-coherent Σ3, resulting in lower diffusivity of the grain boundary network and a decrease in steady state creep rate.

Non-twin Σ3 boundaries included in the grain size measurement procedure are often observed as an integral part of the high angle grain boundary network. As previously stated these Σ3 boundaries have diffusivities 10 to 100 times slower than a RHGB and therefore have the ability to decrease the diffusional flow during Alloy 800H creep. The addition of Σ3 boundaries to the grain boundary network would produce an effective grain size greater than that measured using the current grain size measurement procedure used in the current study. Drabble [115] also recognised the decrease in diffusion for grain boundary engineered Alloy 800H. By employing an electrical resistivity analogy to a simplified 2D grain boundary transport model, Drabble was able to assess the effectiveness of grain boundary engineering strategies for high temperature creep performance.

An additional assumption made in the creep model (Appendix A) is that the entire grain boundary surface is a perfect sink or source for matter during diffusional creep processes. Therefore, the rate of flow is determined only by the rate of diffusive transport from one part of the boundary to another. Arzt et al [160] suggested that the surface as a whole does not act as sink or source for matter, but rather diffusion is controlled by the boundary dislocations. It was shown that the rate of creep depends on the density and mobility of these defects.

Figure 6.23 demonstrates the importance of employing the stereographic correction to provide a more accurate measure of average grain size. If no correction had been used than the green curve would be positioned further to the right given the impression that the creep model is less accurate. The stereographic correction was also a necessity in demonstrating the differences in the frequency of grain at the lower tail of the grain size distribution. With the correction it made it possible to identify variations in lower tail size between samples, for example the difference observed in Figure 6.22.

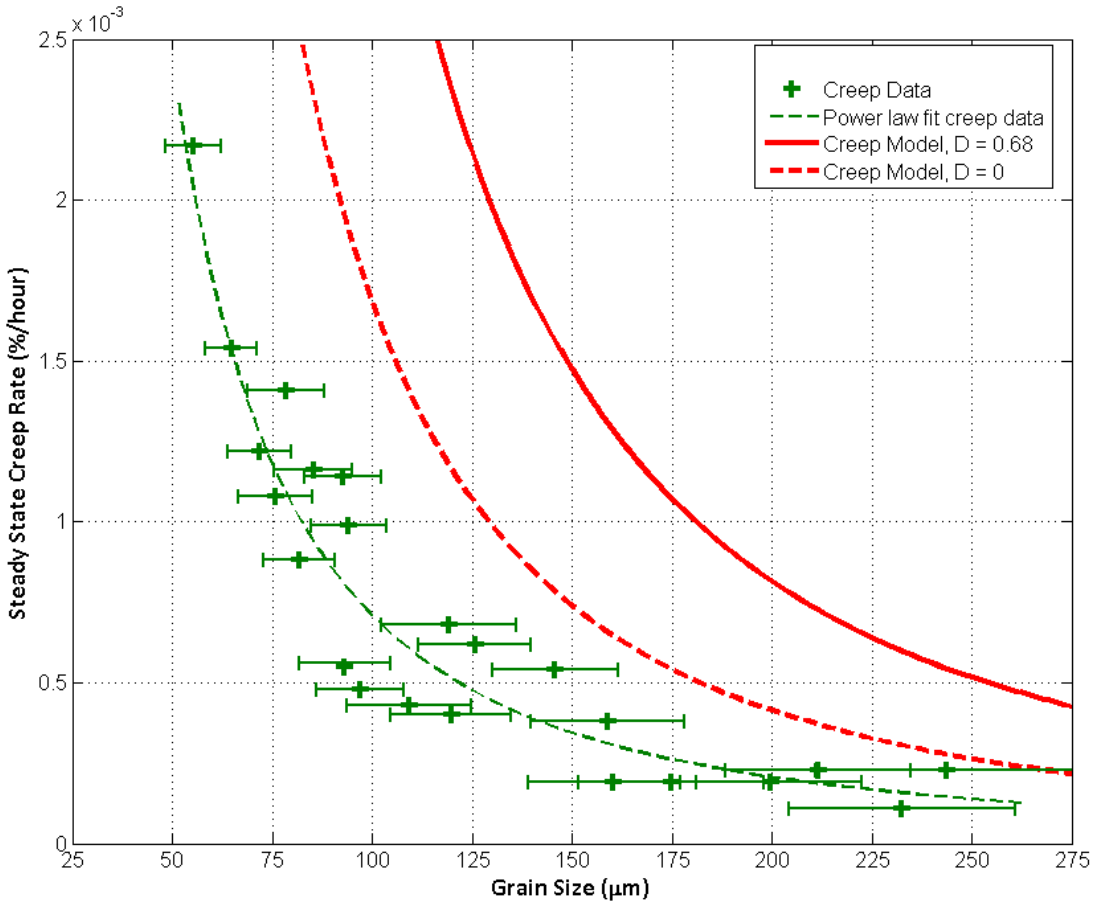


Figure 6.23 Creep rates from predictive models compared to measured creep rates for Alloy 800H.

While it is not possible to directly correlate coherent twin density with steady state creep rate, their effect on creep can be hypothesised by applying understanding of creep mechanisms. Norbygaard [161] studied the effect of Σ3 boundaries on the creep in copper at high temperatures and low stresses. This was achieved by placing a thin grid of Al_2O_3 on the sample surface. The displacement of the grid lines across a grain boundary indicated a grain boundaries participation in the creep process. The study showed that Σ3 boundaries with small v_m (maximum deviation, Equation 2.24) values were inactive during diffusional creep processes, i.e. no grid displacement. Σ3 boundaries with small v_m values are typically assumed to be coherent twins.

The inactivity of coherent twin boundaries during diffusional creep processes comes as little surprise. Minkwitz et al [154] was unable to calculate the diffusion rate of coherent twins and Σ3 boundaries with <110> tilts less than ≈ 15° away from the {111} plane, because grain boundary and volume diffusion zones were so heavily mixed. This result indicates that the diffusivity along a coherent twin (and close tilts) is closer to that of the lattice than a RHGB. This result suggests that a coherent twin would have minimal effect on diffusion creep.

6.3.4 Microstructural Evolution during Creep

Four samples were selected for post-creep analysis to investigate changes in the microstructure, Table 6.4. Two samples, SS1 and SS2, were tested in the G1 apparatus, and two samples, R1 and R2, were tested to failure in the G2 apparatus.

Table 6.4 Four samples selected for post creep test microstructure analysis.

	SS1	**SS2**	**R1**	**R2**
Sample Processing	20%/1150°C	20%/1150°C	20%/1225°C	60%/1200°C
Initial Grain Size (µm)	55.3	64.8	126	97.0
Time at Temperature (hours)	177	385	2632	2632
Steady State Creep Rate (%/hour)	0.00217	0.00154	0.00062	0.00048
Time To Rupture (hours)			1649	2316
Elongation (%)	0.5	2.0	13.8	35.1
Post Creep Grain Size (µm)	58.9	66.8	128	102.4
Change in Grain Shape?	No	No	No	No
Phase ID Performed	×	✓	✓	✓

Creep curves for the four samples are shown in Figures 6.24 and 6.25. The testing of SS1 was terminated after 177 hours, once the sample accumulated approximately 0.5% strain and prior to the onset of tertiary creep. The testing of SS2 was allowed to continue for 385 hours into tertiary creep, resulting in approximately 2% accumulated strain.

Because samples R1 and R2 were tested in the same furnace, both were at temperature for the same duration (2623 hours) even though sample R1 failed approximately 667 hours prior to sample R2[††]. Sample R1 was also the only sample tested in apparatus G2 (rupture rig) not to show a decrease in creep rate during tertiary creep prior to rupture.

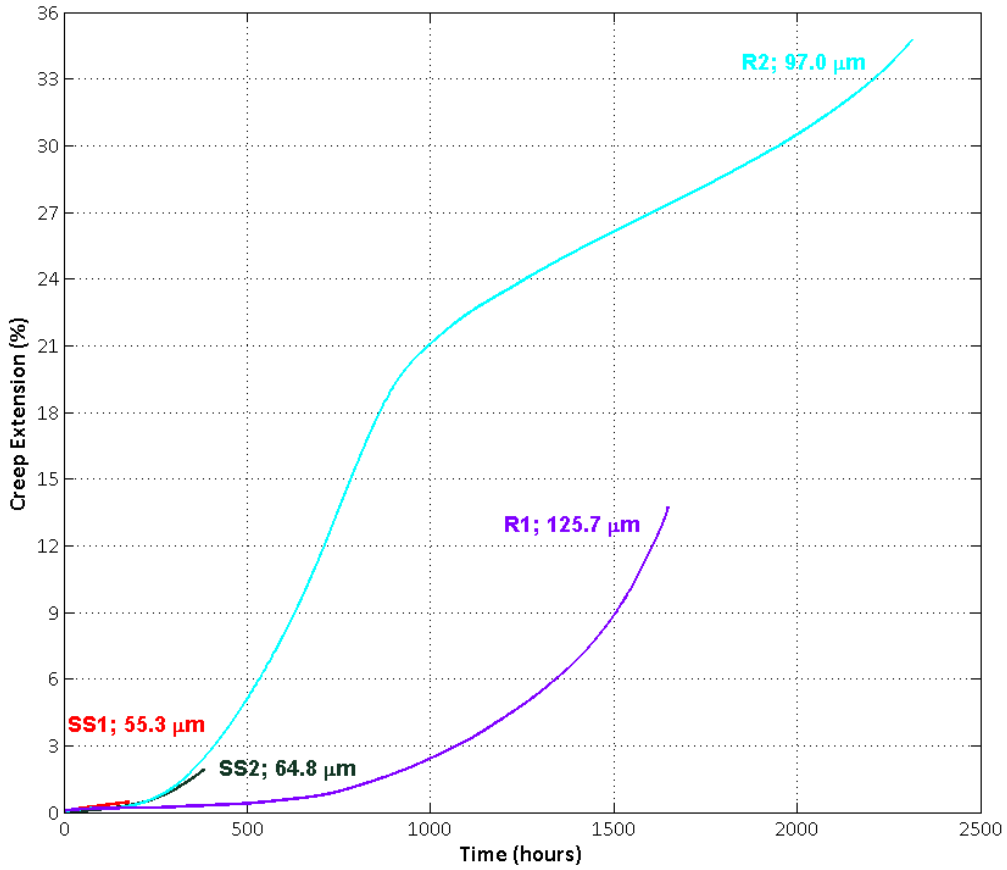


Figure 6.24 Creep curves for the four samples selected for post creep analysis.

[††] Samples could not be removed from the furnace during testing without disrupting other samples.

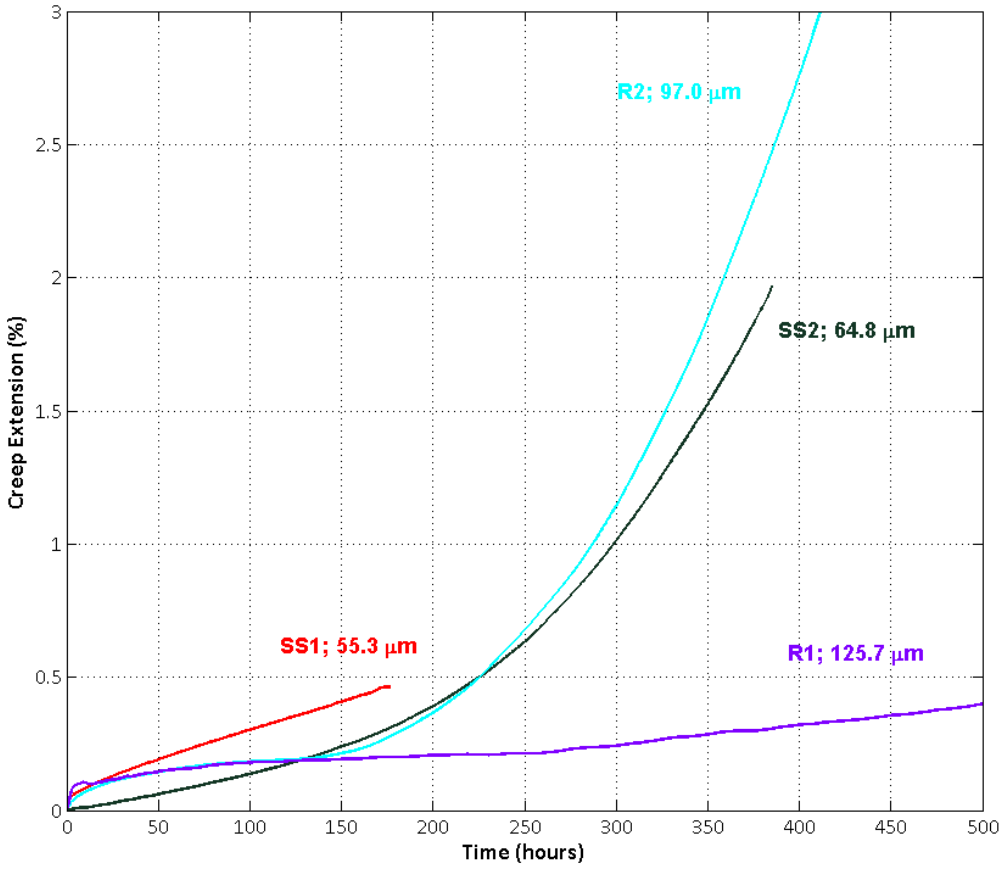


Figure 6.25 Steady state regions of the creep curves for the four samples selected for post creep analysis.

Post-creep EBSD measurement of the average grain size revealed that no significant grain growth occurred during creep. The post creep grain size can be checked by calculating a final grain size using Equation 2.11 and the values n=5.85, k_0=4.82x10^5, and Q = 125 kJ/mol derived in §5.3.6. For example, sample R2 with an initial grain size of 97.0μm was at 980°C (1253K) for 2632 hours (9475200 seconds). The final grain size is given by:

$$\bar{d} = \left(4.82x10^5(9475200)\exp\left(\frac{-125x10^3}{8.314(1253)}\right) + 97.0^{5.85}\right)^{\frac{1}{5.85}}$$

$$\bar{d} = 97.1\mu m$$

Typical grain elongation synonymous with diffusion creep [162] was not observed in the post creep specimens. The lack of grain elongation is consistent with the creep strain measurements, where it was shown that typically less than 0.5% strain was accumulated before the onset of tertiary creep.

Three samples (SS2, R1, and R2) were analysed postcreep using EDS and EBSD to identify phases present in the microstructure. Figure 6.26 shows the backscatter electron (BSE) images for the three samples. The BSE image of sample SS2 shows $M_{23}C_6$ located along grain boundaries. The BSE image of sample R1 shows $M_{23}C_6$ along grain boundaries and Al(CN) within grain interiors. The BSE image of sample R2 shows Cr(CN) along grain boundaries and within grain interiors, and Al(CN) within the grain interiors. Figures 6.27 to 6.29 give representative indexed EBSD patterns, and EDS spectra, for $M_{23}C_6$, Cr(CN), and Al(CN), respectively.

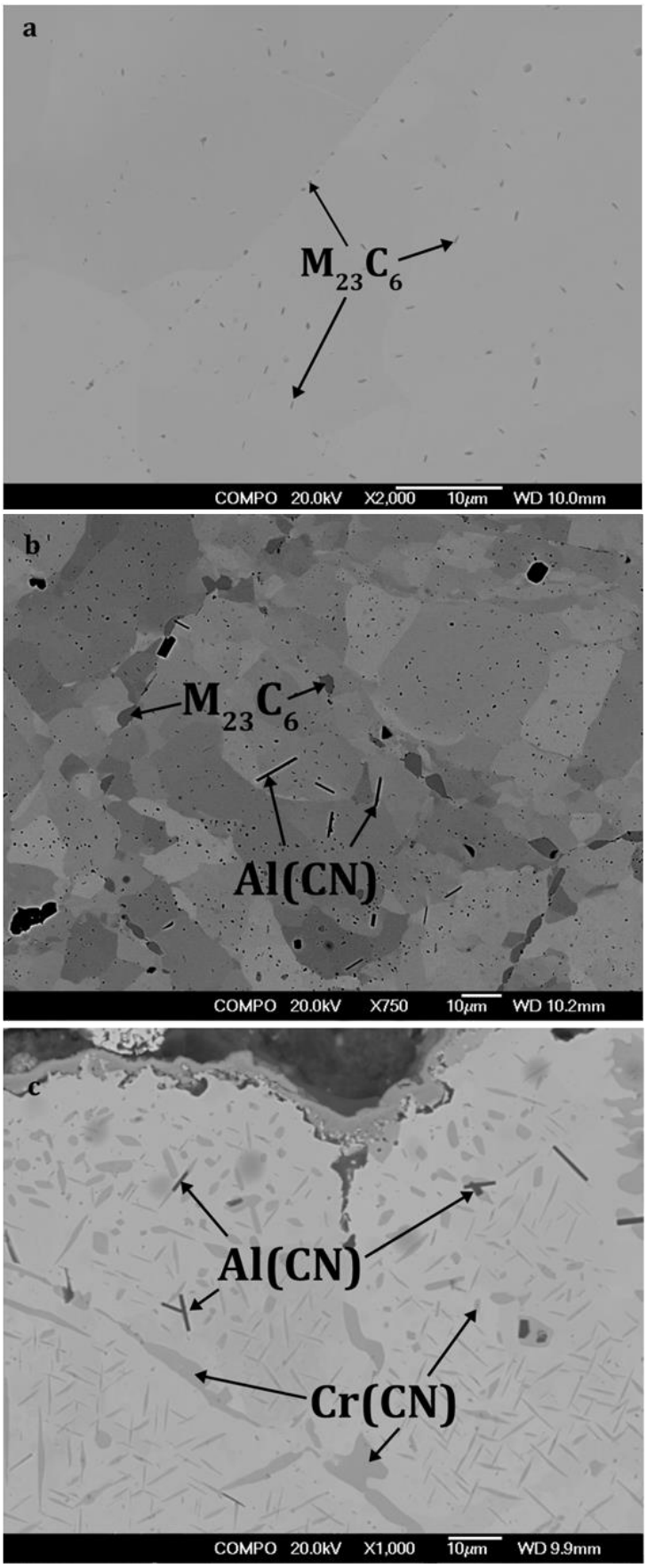
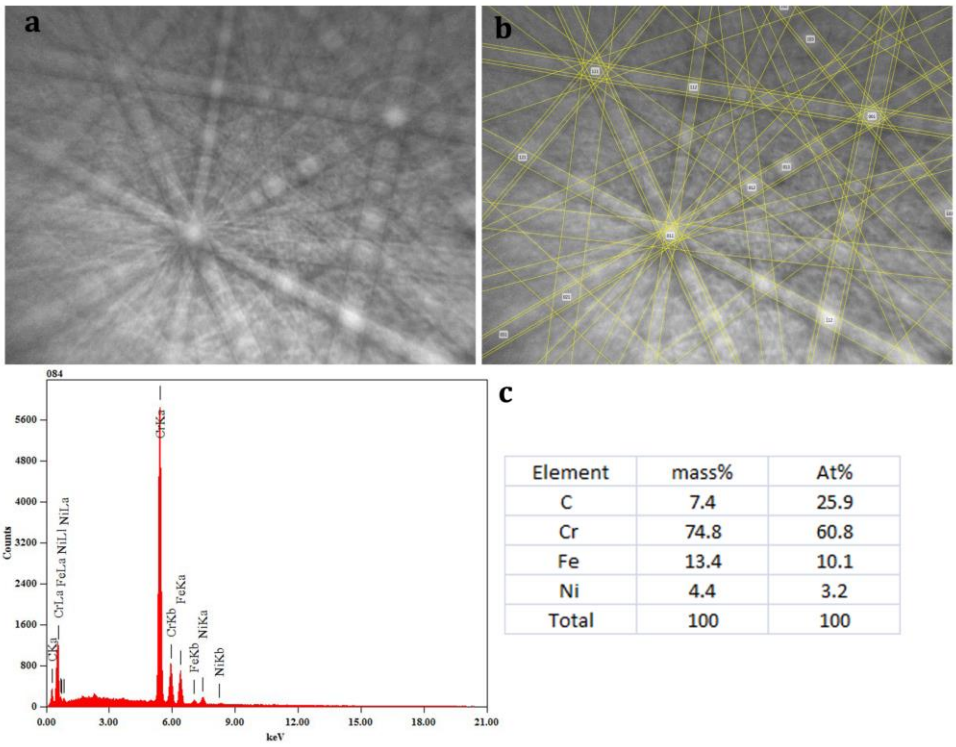
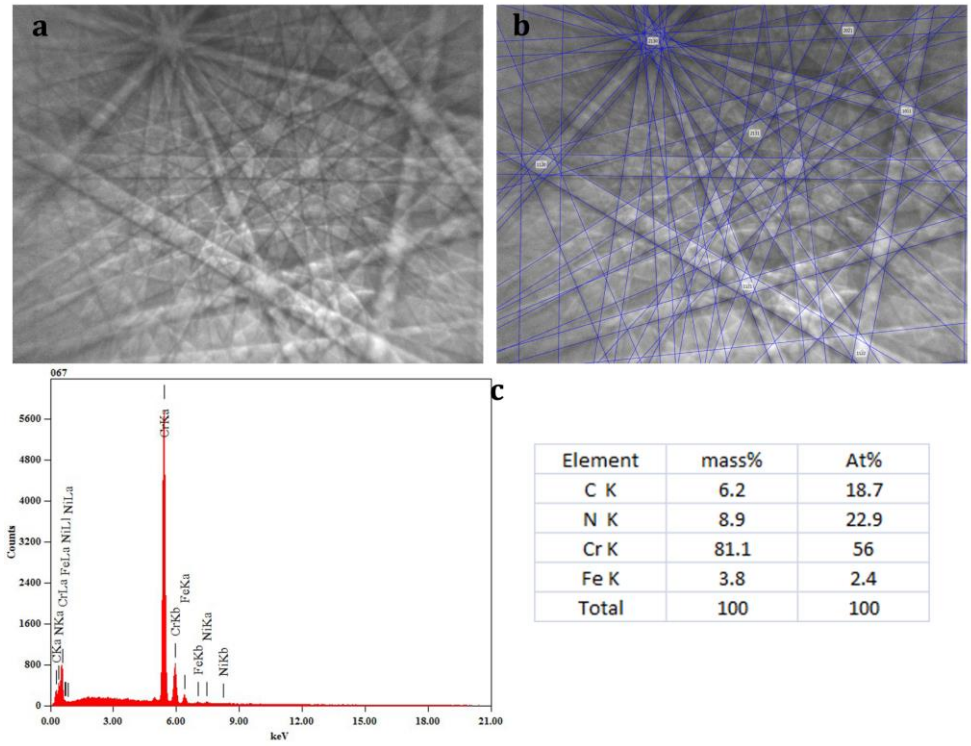


Figure 6.26 Backscatter electron images of samples (a) SS2, (b) R1, and (c) R2.



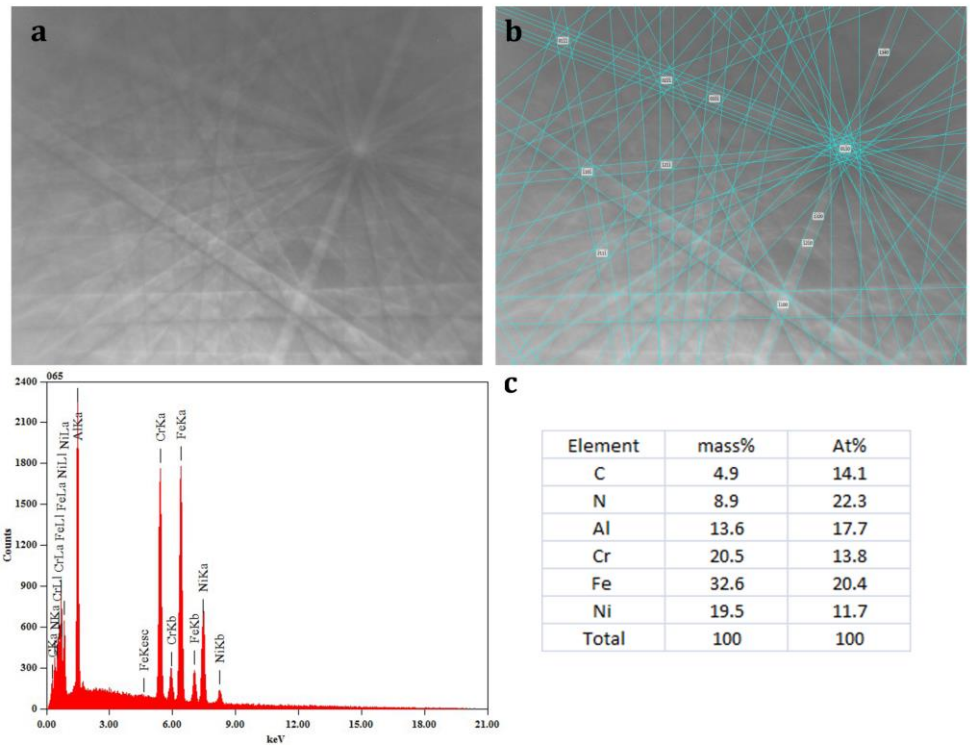
Element	mass%	At%
C	7.4	25.9
Cr	74.8	60.8
Fe	13.4	10.1
Ni	4.4	3.2
Total	100	100

Figure 6.27 (a) Electron backscatter diffraction (EBSP) pattern and (b) the indexed solution for $M_{23}C_6$. (c) EDS spectra and chemical composition of $M_{23}C_6$.



Element	mass%	At%
C K	6.2	18.7
N K	8.9	22.9
Cr K	81.1	56
Fe K	3.8	2.4
Total	100	100

Figure 6.28 (a) Electron backscatter diffraction (EBSP) pattern and (b) the indexed solution for Cr(CN). (c) EDS spectra and chemical composition of Cr(CN).



Element	mass%	At%
C	4.9	14.1
N	8.9	22.3
Al	13.6	17.7
Cr	20.5	13.8
Fe	32.6	20.4
Ni	19.5	11.7
Total	100	100

Figure 6.29 (a) Electron backscatter diffraction (EBSP) pattern and (b) the indexed solution for Al(CN). (c) EDS spectra and chemical composition of Al(CN).

Depending on average grain size, samples were creep tested in the G1 apparatus for between 7 and 28 days. The post creep microstructure of sample SS2, Figure 6.26(a), is indicative of all samples tested in the G1 apparatus. EDS and EBSD identified the precipitates formed during creep as $M_{23}C_6$. Samples R1 and R2, both tested in the G2 apparatus, showed remarkably different microstructures, Figure 6.26. It is important to remember that both samples were at temperature for the same duration (2632 hours), therefore the only difference between them was the length of time load was applied, 1649 hours and 2316 hours for R1 and R2, respectively.

Sample R1, which failed earlier, showed a similar microstructure to that observed for SS2. R1 had $M_{23}C_6$ predominantly on the grain boundaries, and small amounts of Al(CN) was observed within the grains. Erneman et al [9] showed that at 1000°C nitrogen uptake was pronounced in creep deformed material and AlN was present both at the surface and centre of the specimen.

Sample R2 did not present with $M_{23}C_6$ carbides, but rather coarse $Cr_2(CN)$ carbides were present on grain boundaries. Buchanan [156] showed in the aging of Fe-Ni-Cr HP alloys a transformation of $Cr_{23}C_6$ → $Cr_2(CN)$ was observed at 1000°C. $Cr_2(CN)$ precipitates were also present as needle morphology within the grain interiors, along with Al(CN) in quantities greater than that observed in sample R2.

The creep curves shown in Figure 6.24 indicate a reduction in creep rate during the tertiary stage of creep. This was consistent for all samples, except R1, which has been assumed to have failed due to some sample defect. Sample R2 exhibited a reduction in creep rate, while sample R1 failed before any such occurrence. It is therefore suggested that the existence of the large number of nitrides in sample R2 resulted in the strengthening of the microstructure and the corresponding reduction in creep rate.

Why sample R2 presented with large numbers of nitrides compared to R1, even though they were at 980°C for the sample duration, is uncertain. Assuming the diffusion of nitrogen from the atmosphere into the sample would be similar for both samples, questions arise regarding why sample R2 presented with a heavily nitrided microstructure, while R1 had just a few Al(CN) within grain interiors. One possible explanation is that the formation of nitrides is stress-assisted. Tanaka et al [163] showed the nucleation rate of ($Fe_{16}N_2$) precipitates in Fe-N crystals is increased in the presence of a tensile stress. The formation of large numbers of nitrides is not representative of pigtail service conditions, and as a result future accelerated creep testing should be performed in a controlled atmosphere.

6.3.5 Creep Ductility

Figure 6.30 shows the creep curves for the seven samples tested in the G2 apparatus, Table 6.3. One sample was identified as displaying creep behaviour inconsistent with the other samples, possibly due to a sample defect, resulting in premature failure. Another sample had not failed when the test was terminated. All samples (excluding the early failure) displayed the decrease in creep rate during tertiary creep, discussed previously as being associated with the formation of nitrides.

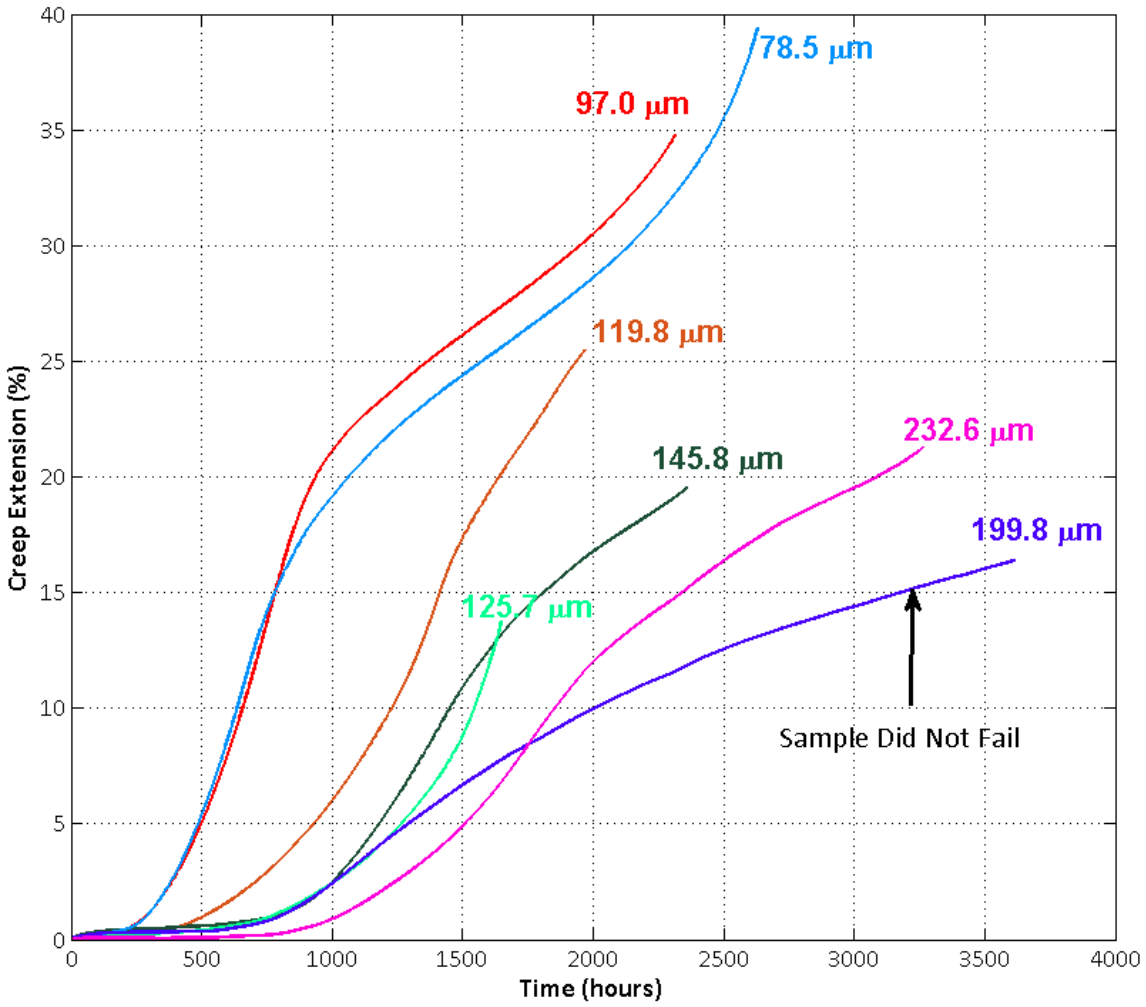


Figure 6.30 Creep curves for samples of different average grain sizes tested in apparatus G2.

Figure 6.31 summarises the main results obtained from the creep rupture testing. The longest rupture times, 3619 hours and 3265 hours, were recorded for the two samples with the largest grain sizes, 199.8µm and 232.6µm, respectively. The shortest rupture time was recorded for the sample (indicated with an arrow) assumed to have failed due to a defect and therefore may be considered an outlier. Unexpectedly, the sample with the smallest grain size, 78.5µm, showed a longer rupture time, 2316 hours, then four samples with larger grain sizes. Without further data it is difficult to draw a definitive conclusion, but tentatively, the results suggest that an increase in average grain size is proportional to the time to rupture. Rupture times for the current testing were up to three times longer than those predicted by special metals [2] (1000 hours) for a testing conditions of 980°C at 13.5MPa.

The bottom plot in Figure 6.31 shows the fraction of time the sample was in tertiary creep. For example, the sample with the average grain size of 78.8µm was in tertiary creep for 93% of 2316 hours (rupture time). The data indicates that as grain size increases, the proportion of time the sample spends in

tertiary creep reduces. The result confirms the observation made previously that samples with larger average grain size have longer steady state regimes.

Finally, Figure 6.31 shows that creep ductility (creep extension at failure, $\%\dot{\varepsilon}_f$) decreases as grain size increases. This result was not unexpected and was discussed Chapter 1. However, the amount of creep strain accumulated was unexpected. Many of the failed Alloy 800H pigtails from Methanex have ductility measured around 10%, 2 to 4 times less than was observed in the current testing. Possible reasons for this difference include accelerated testing conditions such as a temperature approximately 100°C higher than in service, non-uniform grain size distributions observed in pigtails resulting in complex internal stress state, and additional damage mechanisms (stress during start up and shut downs) for the in-service material.

Figure 6.32 shows a weak correlation between steady state creep rate and creep ductility. Further rupture testing is required to determine if a strong correlation exists, although results tend to indicate that samples with larger average grain size will produce slower steady state creep rates and lower ductility.

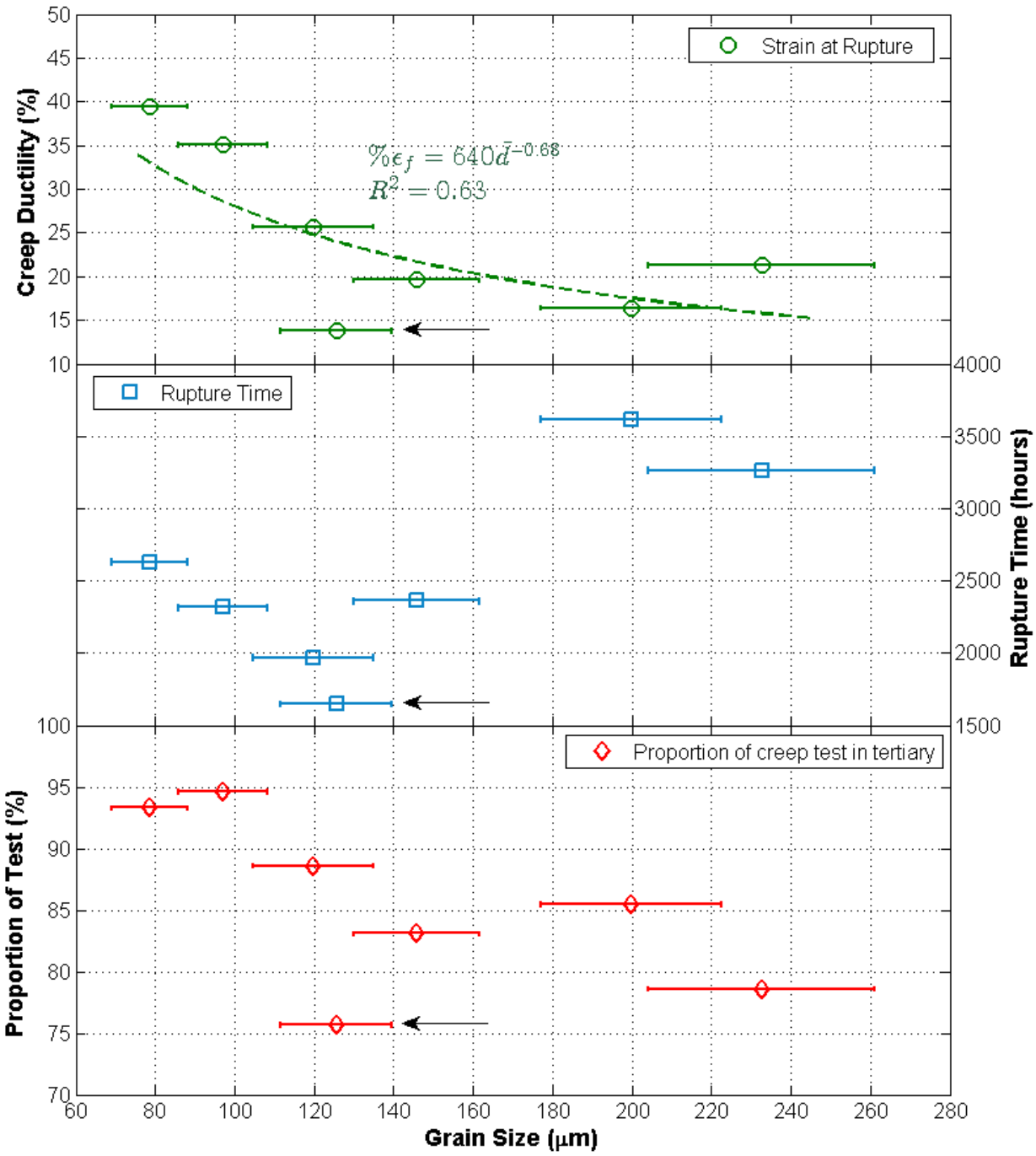


Figure 6.31 Ductility (TOP) and rupture time (MIDDLE) for the seven samples tested to failure. BOTTOM: The fraction of testing time a sample was in the tertiary creep stage.

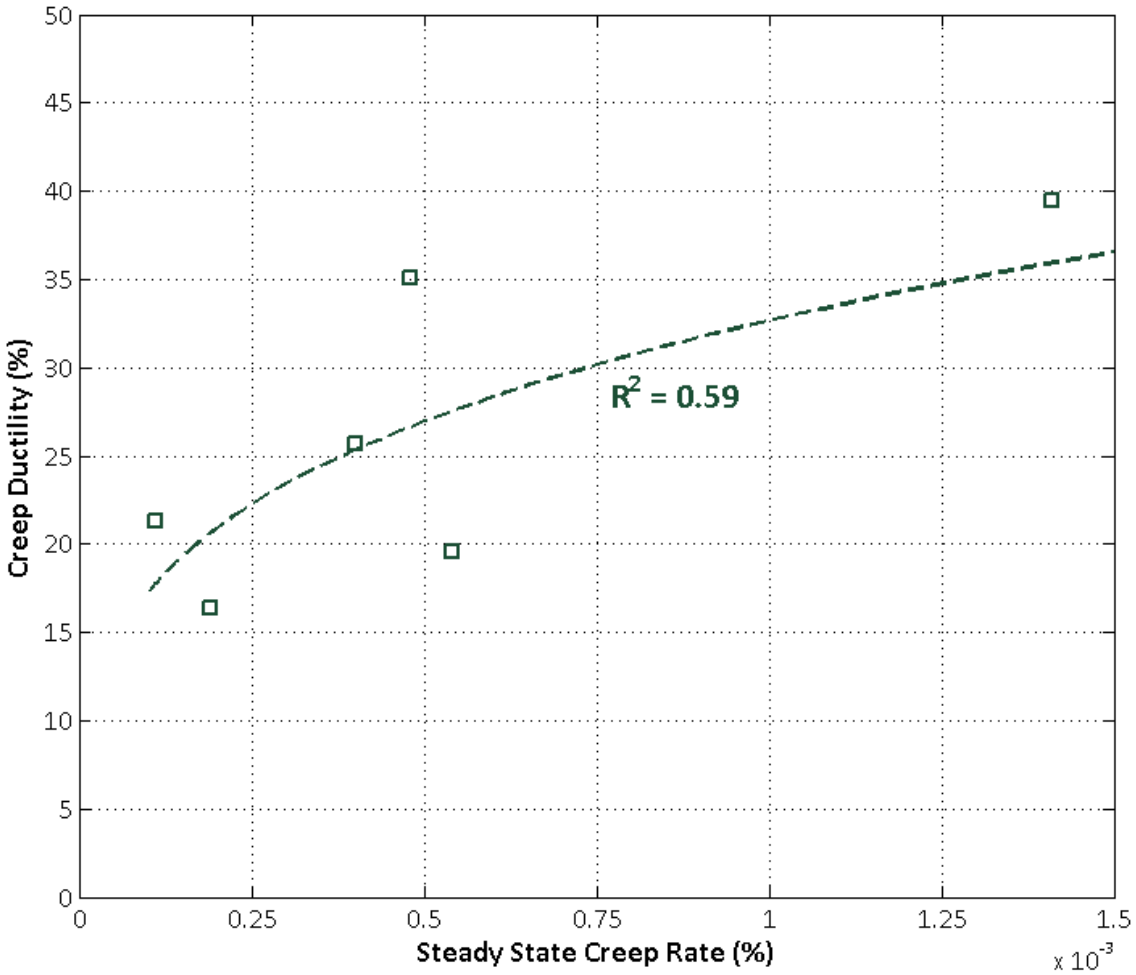


Figure 6.32 Steady state creep rate vs creep ductility for the samples tested to rupture.

6.4 Creep Performance of Alloy 800H Pigtails for Methanex

The focus of the current study was to investigate creep performance and develop a new criterion for the procurement of Alloy 800H for Methanex pigtails. The study investigated three microstructural properties and their effect on steady state creep rate and creep ductility time with varying degrees of success. The microstructural properties investigated were average grain size, grain size distribution, and coherent twin boundaries.

6.4.1 Average Grain Size

The current criterion for the procurement of Alloy 800H considers an average grain size of ASTM grain size number of 5 (≈72μm) or coarser to be appropriate for service. The results summarising the effect of average grain size on creep performance, Table 6.5., shows that there is an order of magnitude difference between the steady state creep rate for an ASTM grain size of 5 (0.00126%/hour) compared to an ASTM grain size of 1 (0.00011%/hour). Additionally, creep ductility can differ by up to 2.5 times between an ASTM grain size of 5 (34.9%) and ASTM grain size 1 (13.6%).

Table 6.5 Summary results for average grain size vs steady state creep rate and creep ductility.

ASTM Grain Size	ECD Grain Size (μm)	Steady State Creep Rate (%/hour)[‡‡]	Creep Ductility (%)[§§]
5	72	0.00126	34.9
4	101	0.00069	27.7
3	143	0.00037	21.9
2	203	0.00020	17.3
1	287	0.00011	13.6

The results indicate that the current acceptable average grain size criterion is too broad and may result in large variations in creep behaviour. It is suggested that Methanex tightens the average grain size criterion to between ASTM grain sizes 2 and 3. Based on the results from the current study this will

[‡‡] Calculated using Power Law fit from Figure 6.20
[§§] Calculated using Power Law fit from Figure 6.31.

ensure a steady state creep rate between approximately 0.00020 and 0.00040%/hour and creep ductility between approximately 17 and 22%.

However, an additional consideration must be made with regard to the feasibility of producing Alloy 800H to this restricted average grain size criterion. From the processing study presented in Chapter 5, it was shown that annealing at 1250°C for between 1 and 2 hours would ensure this range of average grain sizes is achieved.

6.4.2 Grain Size Distribution

The current study only tested samples with uniform grain size distributions having coefficient of variation ranging from 0.764-0.956. Overall, the grain size distribution had very little effect on steady state creep rate. Further rupture testing is required to determine if there may be an effect on ductility. While there is strong evidence to suggest that non-uniform grain size distributions in pigtails have resulted in early failures, this has not been proved in the current study. Chapter 8 details potential future work discussing possible processing techniques for the manufacture of samples with non-uniform grain size distributions and comments on possible methods of testing. Advice to Methanex at this time would be to continue research into pigtail processing methods to prevent the formation of non-uniform grain size distributions.

6.4.3 Grain Size Measurement and Coherent Twin Boundaries

A significant part of the current study was focused on developing a method for Methanex to measure grain size, and determine if coherent twins were to be considered. A significant issue arising from measuring grain size from optical micrographs is that there was no method of identifying coherent twins, apart from morphology. It was shown in the current study that a significant proportion (30 to 50%) of these boundaries may not be coherent twins, but rather <110> asymmetric tilt boundaries. These tilt boundaries may have diffusivities 100 times greater than a coherent twin, and under diffusional creep conditions have behaviour more akin to RHGBs. EBSD mapping combined with trace analysis has been employed to measure average grain size and grain size distribution providing an advantage over optical imagery by including the previously rejected <110> Σ3 tilt boundaries.

Because processing did not allow the formation of microstructures with a significant range of coherent twins, it was not possible to analyse their effect on steady state creep rate. However, previous studies

(Norbygaard [161]) showed that coherent twins were inactive during diffusional creep, while an examination of the diffusivities of Σ3 boundaries with different boundary plane inclinations showed grain boundary and volume diffusion zones heavily mixed. Overall, it is assumed that coherent twins have minimal effect on diffusion creep and do not need consideration.

6.4.4 Relevance of secondary and tertiary creep on the rupture life of Alloy 800H

While a correlation was made between steady-state creep rate and creep ductility (Figure 6.32), it is still important to note that in the samples tested to rupture, only 10% to 20% of total creep life existed in secondary creep and only 1% to 2% of total strain was accumulated during secondary creep. This result would indicate that creep performance is best assessed through creep rupture testing.

It was interesting to observe that all the samples tested to failure in the rupture rig (G2) exceeded the rupture time of 1000 hours given by Special Metals [2] for testing conditions of 980°C and 13.5MPa. This observation was even the case for the sample that failed prematurely (1649 hours) by some assumed defect. When the microstructure of the specimens was analysed post-creep, it showed minimal nitride formation which was attributed to the strengthening shown by the other creep rupture specimens. The result suggests that with careful control over average grain size and grain size distribution, there is potential for Alloy 800H to exceed manufacturer specifications and outperform the 100,000 hour (11.4 year) design life for pigtails.

6.5 Summary

Chapter 6 detailed the creep performance of 23 Alloy 800H creep specimens produced with a range of average grain sizes, 61.8-243.7µm, coefficient of variation, 0.764-0.956, and coherent twin length fractions, 24.6% to 31.5%. Creep performance was assessed primarily by measuring the steady state creep rate and creep ductility.

The results showed that samples with larger average grain size had longer steady state creep regimes, 125 and 700 hours for average grain sizes of 64.8µm and 232.6µm, respectively. Also observed was the minimal amount of strain accumulated before the onset of tertiary creep, less than 0.6% extension representing approximately 2% of the total accumulated strain at rupture. The steady-state creep rate showed a grain size dependence of $\bar{d}^{-1.78}$, indicating that diffusional creep processes dominated. Samples with tighter grain size distributions typically displayed lower steady-state creep rates due to fewer small grains. However, the difference shown was minimal, and coupled with the observation that steady state creep accounts for at most 0.6% of the total strain, the effect on overall creep life is inconsequential.

The steady-state creep rates were compared to the values calculated from the model (Appendix A). The model showed steady state creep rates larger than those measured in the current study. This disparity was predominately due to the grain size measurement strategy employed in the current study. The model assumed all boundaries had equal diffusivities, while in the current study non-coherent Σ3 boundaries were included in grain size measurement. These Σ3 boundaries included <110> asymmetric tilt boundaries with morphologies often identical to coherent twins and therefore omitted during grain size measurement from optical micrographs. The diffusivities of the non-coherent Σ3 boundaries are 10 to 100 times slower than a RHGB, a range still closer to the diffusivity of a RHGB than that of a coherent twin. Coherent twins have diffusivity closer to that of the lattice. Typically, the total boundary length included in grain size measurement contained 25 to 30% non-coherent Σ3, resulting in lower diffusivity of the grain boundary network and a decrease in steady state creep rate.

Three samples (SS2, R1, and R2) were analysed post-creep using EDS and EBSD to identify phases present in the microstructure. Two samples tested to rupture showed remarkably different microstructures even though they were at temperature for the same duration (2632 hours). The only difference between them was the length of time where load was applied, 1649 hours and 2316 hours. The sample that ruptured after 1649 hours had $M_{23}C_6$ predominantly on the grain boundaries, and small amounts of Al(CN) was observed within the grains, while the sample that ruptured after 2316 hours had coarse Cr_2(CN) carbides present on grain boundaries and as a needle morphology within the grain

interiors. Al(CN) was also present within grains in quantities greater than that observed in the other sample. The sample that ruptured after 2316 hours exhibited a reduction in creep rate during tertiary creep, while other samples failed before any such occurrence. It is suggested that the existence of the large number of nitrides in sample R2 resulted in the strengthening of the microstructure, and the corresponding reduction in creep rate.

Because only seven samples were tested to rupture, few strong conclusions can be made regarding the effect of grain size and grain size distribution on creep ductility. Creep ductility (creep extension at failure, $\%\dot{\varepsilon}_f$) decreases as grain size increases, 39.5% for an average grain size of 78.5µm and 21.3% for an average grain size of 232.6µm. Overall this result was not unexpected, what was unexpected was the amount of creep strain accumulated in the samples. Many of the failed Alloy 800H pigtails from Methanex have ductility measured around 10%, 2 to 4 times less than observed in the current testing.

Although there remains scope for additional creep testing, the current results have provided valuable information for Methanex to develop a revised Alloy 800H procurement criterion. The current 'ASTM 5 or coarser' criterion may produce an order of magnitude difference for steady state creep rates, and 2.5 times difference for creep ductility. It is suggested to Methanex that an ASTM grain size of between 2 and 3 is used. Unfortunately, no data was collected regarding the difference in creep rates between samples with uniform and non-uniform grain size distributions. Uniform and non-uniform grain size distributions are often observed in pigtails and, as previously suggested, may be a factor in early failures. Finally, it was observed that the material exceeded the 1000 hour rupture time given by Special Metals [2] for testing conditions of 980°C and 13.5MPa. Even the samples assumed to have failed prematurely produce a rupture time greater than 1.5 times the 1000 hour rupture time.

Chapter 7 Conclusions

In the current study EBSD mapping and trace analysis was used to identify boundary types. However, unlike many studies employing similar analysis methods, the current investigation presents an extensive examination on the influence of EBSD mapping parameters on the identification of grain boundary elements and the resulting quantification of the grain boundary character distribution. The study concludes:

- When the mapping step size approaches the distance (width) between a pair of parallel Σ3 boundaries, either fragmentation occurs or the boundary is missed entirely.
- The Σ3 boundary's fragmented appearance results in over-quantifying Σ3 boundaries and under-quantifying RHGBs.
- It was shown that grain boundary fragmentation has minimal effect on the length fraction. In comparison, an error of at least 15% on the number fraction results from the boundary fragmentation.

Grain size measurement was performed using either linear intercept or flood-fill (equivalent circle diameter) methodologies on both optical micrographs and EBSD boundary maps. The results concluded:

- The twins excluded intercept distance was 101µm for optical and 77µm for EBSD respectively.
- The difference indicates that only approximately half of the coherent twin boundaries identified optically by their morphology were later confirmed through EBSD and trace analysis. This result reinforced the initial assertion that coherent twin boundaries cannot be identified simply by analysing their 2D morphology.

The measurement of 3D grain size from a serial sectioned Alloy 800H volume was used to validate the Saltykov stereographic correction for grain size distributions measured in 2D. The analysis concluded:

- Saltykov correction increased the number of grains at the upper tail of the distribution.
- The shift in the distribution increased the average grain size from 51.5µm to 61.0µm, which is within 6% of the average 3D grain size measured from the reconstructed EBSD serial sections of 64.9µm.

A significant contribution has been made to the current field of research through the analysis performed on the 3D reconstructions of twin volumes and twin interfaces. From the current study it was concluded:

- The morphologies represented in 2D provided a poor representation of a volume's true complexity. Intersecting $\Sigma 3^n$ boundaries produced 3D morphologies with multiple re-entrant geometries unable to be reliably represented on a single 2D section.
- The resulting 3D reconstructions allowed for the determination of the grain boundary plane inclination, and, along with the orientation data, all five DOF which define a grain boundary were determined. From this analysis it was observed that Σ3 boundaries containing broad planar interface area were not always seen to be coherent.
- Faceting was observed in the majority of the reconstructed Σ3 interfaces. As a consequence of the sectioning depth required to produce a reasonable volume, z-resolution was not fine enough to reveal these relatively small features.
- A theory was proposed here as to why boundary faceting occurs, which suggests that the formation of a Σ3 asymmetric tilt boundary is favourable initially during recrystallization, and then dissociates into a boundary with an interface comprised of coherent ledges and non-coherent risers if energetically favourable to do so.

Chapter 5 presented an investigation analysing the effect of thermo-mechanical processing on the grain boundary character distribution of the high-angle grain boundary network of Alloy 800H. Although the literature documents similar investigations for numerous alloy systems, we believe that the current analysis on Alloy 800H provided a unique contribution by providing additional analysis on the coherent nature of Σ3 boundaries. The main conclusions from the study include:

- No appreciable Σ3 formation occurs during grain growth.
- Temperatures between 1200°C and 1250°C had minimal effect on the formation of Σ3 boundaries for Alloy 800H.
- Samples produced with 6% cold-work followed by annealing at 1200°C displayed some of the largest Σ3 length fractions, 53.9%. This was due to the sufficiently low strain promoting the migration of existing high-angle grain boundaries modifying rather than replacing the existing microstructure.
- A decrease in Σ3 length fraction with increasing strain is the result of an increase in driving force for the nucleation and growth of new grains during recrystallization. This idea is confirmed by

producing samples with 80% cold-work followed by annealing at 1350°C resulting in Σ3 length fractions of 42.5%

- Although the samples prepared with 20% cold-work followed by annealing at 1000°C produced the largest overall Σ3 length fractions from all samples, 57.4%, it did not result in a similar increase in coherent Σ3 length fraction, 30%. The non-coherent Σ3 boundaries formed due to processing appeared curved and are likely to have high mobility. Additional annealing resulted in a decrease in Σ3 length fraction, 50.2%.

The relationship between grain size and creep performance at a stress and temperature reflective of pigtail operating conditions has not been observed in any publication or study. We feel that this information provides a major contribution to the literature and will prove invaluable for designers of Alloy 800H components for high temperature applications. The major conclusions from the creep testing include:

- The steady-state creep rate showed a grain size dependence of $\bar{d}^{-1.78}$, indicating that diffusional creep processes dominated.
- Samples with tighter grain size distributions typically displayed lower steady-state creep rates due to fewer small grains. However, the difference shown was minimal, and coupled with the observation that steady state creep accounts for at most 0.6% of the total strain, the effect on overall creep life is inconsequential.
- Nitride formation greater than that observed in post-service pigtails was observed in rupture samples. The formation of nitrides was due to a testing temperature approximately 100°C higher than pigtail service temperatures.
- The formation of nitrides was observed to occur at higher strains resulting in a decrease in creep rate during tertiary creep.
- The current 'ASTM 5 or coarser' criterion may produce an order of magnitude difference for steady state creep rates, and 2.5 times difference for creep ductility. It is suggested to Methanex that an ASTM grain size of between 2 and 3 is used.
- By maintaining a uniform grain size distribution rupture times of 1.5 to 3.5 times the manufactures' data for Alloy 800H are obtainable.

Note that no data was collected regarding the difference in creep rates between samples with uniform and non-uniform grain size distributions (see future work).

Chapter 8 Future Work

8.1 Introduction

The present research has addressed the initial scope of study. However, it has also opened some further avenues of exploration which may be of scientific or industrial importance. This section details the major areas in which further research may prove useful.

8.2 Identification of Interface Planes of Σ3 Facets

The serial sectioning and 3D reconstructions of Σ3 crystal volumes revealed complex morphologies of which 2D sectioning was unable to fully envisage. Few Σ3 boundaries were shown to consist of a single planar interface. Typically these boundaries were constructed of broad coherent ({111}) interface planes with smaller interface facets (risers). The serial sectioning thickness and resulting resolution of the 3D reconstructions made it difficult to identify the interface planes of these facets, although research on copper [65] have shown many to be orientated 82° away from the {111} coherent twinning plane around the <110> axis. This orientation, often called the 9R structure, was shown by Wolf et al [63] to display a dip in energy, Figure 8.1.

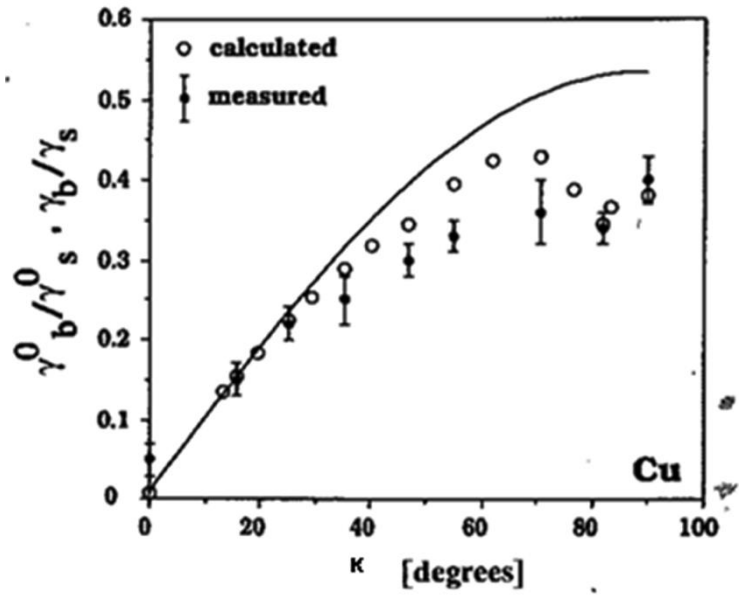


Figure 8.1 Grain boundary energy, γ_b, versus inclination for Cu Σ3s. γ_s is the {110} surface energy of copper [63].

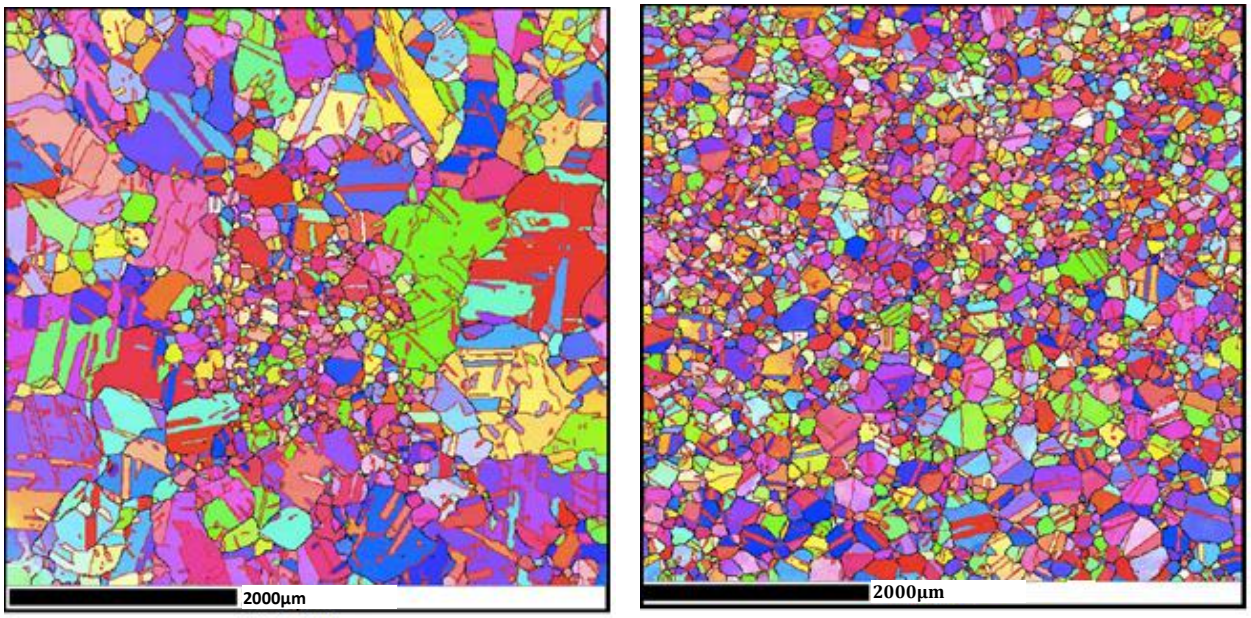
To improve understanding of the interface structure of Σ3 boundaries, targeted sectioning and 3D reconstruction of specific boundaries is suggested for future work. A complete analysis of the area and inclination of the Σ3 boundary interface planes will provide additional information related to properties, i.e. energy and diffusivity, and how this may influence creep performance.

The feasibility of serial sectioning and analysis of interface facets has been investigated. The use of a FIB-SEM was determined to be the obvious first choice for improving the resolution of the 3D reconstructions. A 50x50x50µm volume of material with an EBSD step size of 0.2µm would produce an x-y resolution similar to that seen in the present reconstructions, but with a proposed sectioning depth of 0.2µm a 10-times improvement in z-resolution is possible. A major difficulty with the proposed study is preparing a suitable area of sample for analysis. Due to the limited size of material that can be analysed with FIB-SEM sectioning it is vital that the selected region produce the relevant information. You dont need to make excuses.

8.3 Pigtail Representative Grain Size Distributions

The thermo-mechanical processing used in the current study to vary grain size and grain size distributions did not produce microstructures representative of some post-service Alloy 800H pigtails. Figure 8.2 shows EBSD orientation maps from two pigtails demonstrating the variations that exist in the grain size distributions. The microstructure from the 'Sumitomo' pigtail has a grain size distribution typical of that produced through the processing employed in the current study. The microstructure from the 'Chile' pigtail has a region of small grains in the centre surrounded by larger grains. Preparing and creep samples with microstructures representative of the Chile pigtail may be relevant for furthering the understanding of creep in some Methanex pigtails.

Asymmetric rolling is one method that may be employed to produce non-uniform grain size distributions. Asymmetric rolling involves the angular velocities of the upper and lower rolls to be different so that shear deformation is applied throughout the thickness of the sample. When annealed, different volumes within a sample will experience varied recrystallization rates with the result being a non-uniform grain size distribution.



Chile (F4P2) **Sumitomo**

Figure 8.2 Varied pigtail grain size distributions.

Additional consideration must also be given to the preparation of the creep samples. Care must be taken to cut the sample from the strip of material so that the microstructure is representative of the pigtail microstructure. This may be difficult with the current sample dimensions of 4mmx3mm cross-sections, particularly any creep response that may occur if a large grain is located over the entire sample thickness. Pipe rupture tests, whereby test pieces are the same wall thickness of an actual pigtails, is one option to mitigate this issue.

8.4 Producing Microstructures with Varied Twin Fractions

One of the current project objectives was to investigate the effect of twin boundaries on creep performance. Unfortunately, Alloy 800H was not the ideal system for this type of investigation due to the difficulty of preparing samples with varying coherent twin boundary length fractions. A minimum length fraction of 25% was achieved and while it was possible to produce length fractions greater than 30% using GBE processing techniques, this would typically result in the break-up of the RHGB network.

Figure 8.3 shows the average Σ3 frequency for various material systems for over 200 investigations [116]. The results show that fcc steels typically have high (≈30%) Σ3 boundary frequencies. Lehockey and Palumbo [164] managed to produced two samples of nickel (99.99%) with different Σ3 boundary frequencies. Sample 1 had a grain size of 35µm and a Σ3 fraction 27.9% and sample 2 had a grain size of

25µm and a Σ3 fraction of 46.7%. The two samples were prepared from cast material having an average grain size of 50µm and a Σ3 length fraction of 1.6%. The ability to produce samples from nickel with Σ3 boundary fractions lower than anything produced in the current study (less than 40%) is because of nickel's higher stacking fault energy, 120-130mJm^{-2} [165], compared to austenitic stainless steel that typically have stacking fault energies less than 50mJm^{-2} [166]. Using materials with higher stacking fault energies may be one way of producing samples with a larger range of Σ3 boundary fractions.

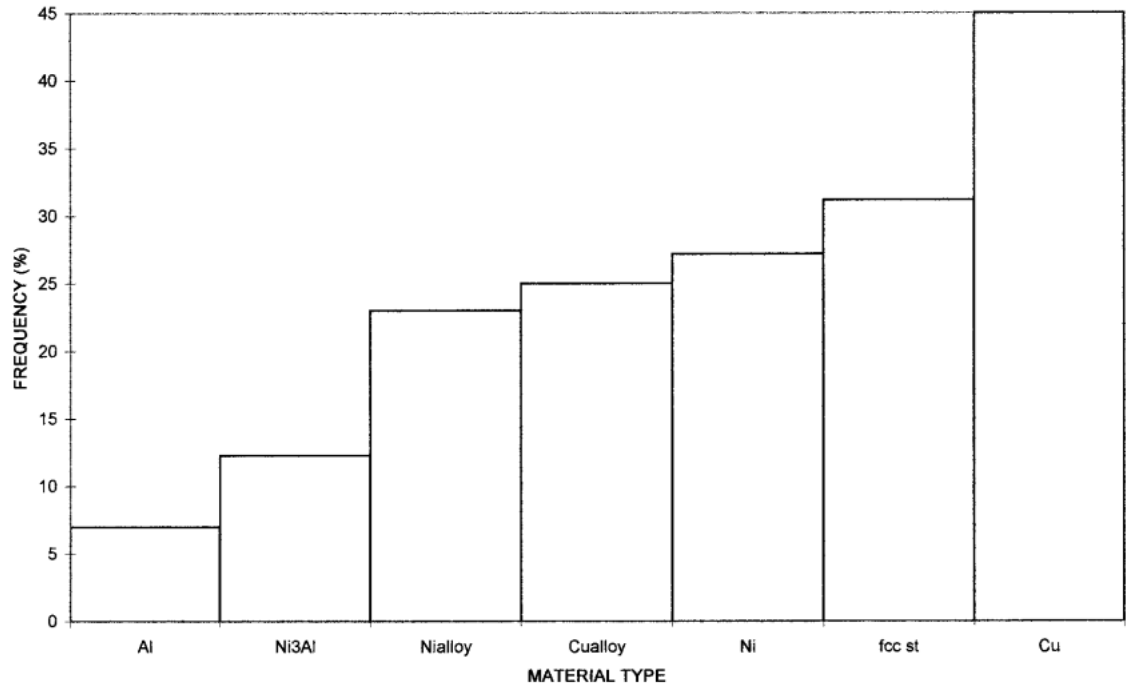


Figure 8.3 Frequency of Σ3 boundaries in various materials (fcc st = f.c.c steels) [116].

8.5 Evidence for Creep Mechanisms in the Alloy 800H

An opportunity exists for an experiment analysing the change in the Alloy 800H microstructure during creep. The proposed experiment would be an interrupted creep test analysing the samples for the duration of their creep life (minimum of four months). The creep test would be periodically stopped at planned intervals and the microstructure analysed to identify changes in grain size, grain shape, internal deformation state, grain boundary positions, evidence of voids and cracks due to creep, and the formation of carbide and nitride phases. The proposed experiments would be performed at various temperatures (i.e. 880°C to 980°C) and stresses (i.e. 10MPa to 20MPa) to determine if there is any evidence of a change in creep mechanisms assessing the validity of accelerated creep testing.

A carefully designed experiment would be able to give valuable insights into the following areas:

1. Evidence for the secondary creep mechanisms in Alloy 800H.
2. Evidence for the tertiary creep mechanisms in Alloy 800H.
3. Process that controls the transition from secondary to tertiary creep in Alloy 800H.
4. Evolution of secondary phases during creep.
5. The relevance of accelerated creep testing conditions to Methanex operating conditions.

In the current study, the evidence for identifying diffusional processed as the dominated secondary creep mechanism was the grain size dependence on steady state creep rate. While the data showed as strong grain size dependence (Figure 6.20), other grain size sensitive mechanisms, such as grain boundary sliding, may be operating. By performing an interrupted creep tests and analysing the same area on the sample surface, a time lapse of the microstructure evolution [167] can be constructed to assess changes in grain size, grain shape, grain boundary positions, and the internal deformation state of the microstructure. Combined with the elongation vs time creep curve, the change in microstructure would provide compelling evidence as to the dominate secondary creep mechanism operating at certain conditions.

The same interrupted creep experiment can also be employed to understand the damage mechanisms associated with tertiary creep. In the current study tertiary creep was associated with an increase in creep rate. The experiment would focus principally on the change in microstructure during the transition from secondary to tertiary creep identified as the inflection on the creep curve. It is during this period we would expect to see an increase in the number and size of voids along grain boundaries and at grain corners. Eventually we would expect to see these voids coalescence and form cracks. This study would also assist in identifying the preferential locations of voids and cracks. It was previously observed and hypothesised that cracking occurs preferentially around the largest grains in the microstructure and an experiment such as the one proposed may add clarity to these assumptions.

The post-test microstructures analysed in the current study showed significant formations of nitrides typically not associated with Methanex pigtail operating conditions. The nitride formation was associated to the decrease in creep rate observed during tertiary creep. The proposed interrupted creep study would also be used to analyse the formation of secondary phases throughout creep life and the effect it has on creep rate. By performing multiple tests at different temperature-load combinations it would be possible to identify the conditions necessary for the different phase formations.

8.6 Modifications to Creep Rig

High temperature creep testing has been performed at the University of Canterbury for approximately 10 years. The first generation of creep testing machines introduced isothermal furnace liners to ensure temperature uniformity and allow the simultaneous testing of up to four samples in series. The second generation of creep testing machines allowed the loading of individual samples and testing to rupture. Several additional modifications are planned for future generations of creep rigs:

1. Load cells to ensure accurate load measurements.
2. Introduction of argon atmosphere to reduce the formation of nitrides at elevated temperatures.
3. Temperature controlled room to limit the effects of ambient temperature fluctuations.

In the second generation creep rig lever arms were used to apply the test load to each sample, Figure 8.4. Each arm was designed to rotate on a knife edge pivot point and apply a ten-fold load magnification factor to the sample. It is also noted that the self-weight of the lever arm was designed to place a 5kg load on each sample. This self-weight provided the preload while heating the furnace to the testing temperature (as directed by ASTM E139).

The rotation of the lever arm due to the sample's elongation during testing and the wear on the knife edge at the pivot point may affect the load transferred to the sample. A proposed modification is to attach load cells to the bottom tensile load bars. These load cells can accurately measure the tensile load transferred from the load arm to the sample and monitor any changes that may occur during testing.

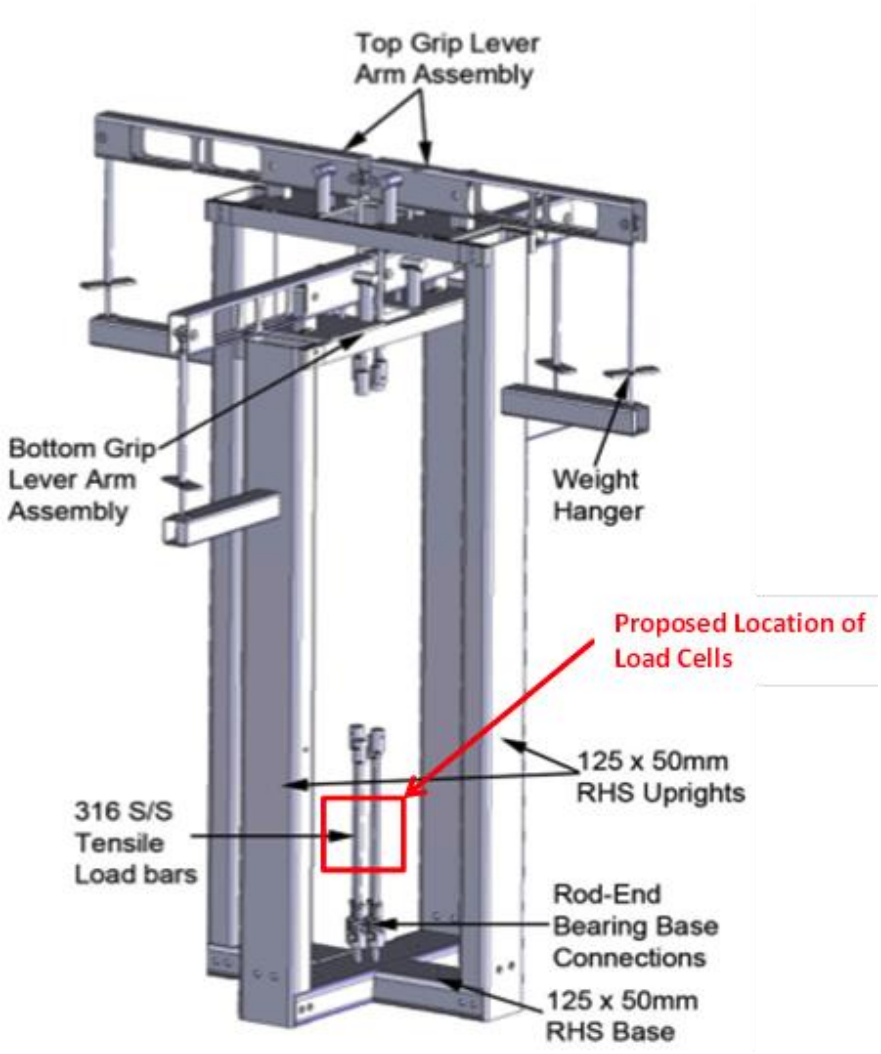


Figure 8.4 CAD drawing of the frame of the second generation creep apparatus indicating the four load arms for individual loading of the four samples and the proposed location of the load cells.

Results from rupture testing showed significant formation of nitrides, particularly after large creep strains, ε>15%. The production of aluminium and carbon nitrides has been shown [9] to occur at the current testing temperatures (approx. 1000°C) in volumes greater than that seen in post-service pigtails. To limit the formation of nitrides it is proposed that future testing use a continuous flow of argon to fill the furnace space restricting the quantity of air present.

The current creep testing environment is a publically used space with no climate control. The current setup results in fluctuations in ambient temperature resulting in the creep rig frame temperatures variations of ±5°C. The fluctuations in ambient temperature were plotted alongside LVDT extension for a period of 96 hours. The ambient temperature was recorded approximately 300mm from the LVDT with results showing that LVDT extension is affected by the variations in ambient temperature.

Designs for the rebuilding of the current engineering lab wings at the University of Canterbury (construction proposed for 2015) have allowed for the construction of a temperature controlled room for creep testing. This room is not accessible from any of the public walkways and has no windows meaning drafts and vibrations are minimised. Heat pumps will be installed to stabilize temperature.

8.7 Synthetic Microstructure Generation and Creep of Alloy 800H

The current study was able to produce valuable information regarding the three-dimensional morphology of twinned microstructures and the effect of thermo-mechanical processing on grain size statistics, grain boundary distributions, and grain boundary network topology. Progress was also made in understanding the relationship between the microstructures produced through processing and high temperature creep response. In addition to the technical challenges of high temperature creep testing, the costs, and length of time required to perform such testing, one of the major downsides to this strictly experimental approach is the difficulty to vary one microstructural feature while holding others constant. This is due to the evolution of most of these microstructural features being highly coupled during processing.

As discussed above the development of microstructures with varied twin densities, average grain size, and grain size distributions was difficult. Therefore, another method of determining microstructure-property relationships is through the development of synthetic microstructures and computer modelling creep response. One of the major beneficial aspects of this approach is that a large array of synthetic samples with different microstructural features can be generated and tested at a low cost in a shorter time frame compared with the testing of real samples. Furthermore, in contrast to the limitations when generating real samples, it is possible to vary the microstructural features of these synthetic samples independently in order to gain a more direct understanding of their effect on properties.

Different strategies have been employed to generate 3D synthetic microstructures in order to examine microstructure-property relationships. Saylor et al [168] and Brahme et al [169] used experimental grain size, shape and orientation statistics in 2D to infer 3D grain structures, while Lewis et al [170] used the volumes reconstructed from serial sectioning of AL-6XN, an austenitic stainless steel, to generate three-dimensional meshes. Image based finite element simulations were performed to analyse the elastic response at grain boundary junctions comprised of different boundary types. Tucker et al [171] discussed the formation of synthetic microstructures with grain size distributions of log-normal types and twin boundaries. The twin insertion algorithm employed for developing the synthetic

microstructures assumes that the twin boundary always completely sections the parent grain. The 3D reconstructions of Alloy 800H in the current study showed that Σ3 boundaries can be significantly more complicated and may have interface planes other that the {111} coherent twin plane. The statistical and morphological information amassed in the current study will be valuable to the generation of synthetic microstructures representing Alloy 800H microstructures.

Appendix A Modelling Steady-State Creep Rate

The following model uses diffusion-based rate equations to predict the steady state creep rate for samples of various average grain size and grain size distributions.

A.1 Modelling Power Law Creep

At the temperatures and stresses of interest, both glide and climb may participate in the deformation process. Frost and Ashby [11] proposed the shear strain rate, $\dot{\gamma}_{PL}$, for power-law creep as

$$\dot{\gamma}_{PL} = \frac{AD_{eff,PL}\mu b}{k\,T}\left(\frac{\sigma_s}{\mu}\right)^n$$

Equation A **.8.1**

where $D_{eff,PL}$ is the effective diffusivity composed of the relative contributions from lattice and dislocation core diffusion

$$D_{eff,PL} = D_v\left[1 + \frac{10\,a_c}{b^2}\left(\frac{\sigma_s}{\mu}\right)^2\frac{D_c}{D_v}\right]$$

Equation A. 8.2

where

$\dot{\gamma}_{PL}$	shear strain rate [s^{-1}]
A	dimensionless constant
μ	shear modulus [Pa]
T	temperature [K]
σ_s	shear stress [Pa]
n	power-law creep stress exponent

$b = 2.58 \times 10^{-10}\ m$	Burgers vector (Austenite [11])
$k = 1.381 \times 10^{-23}\ m^2\ kg\ s^{-2}\ K^{-1}$	Boltzmann constant
D_v	lattice diffusivity [m^2/s]
D_c	dislocation core diffusivity [m^2/s]
a_c	dislocation core cross-sectional area [m^2]

The shear modulus is a temperature dependent property. Using the published data from the Special Metals publication [2] the change in shear modulus, μ, with respect to temperature, T, is derived

$$\mu = -3x10^7T + 8x10^{10}$$

Equation 8.3

A.2 Determining Power Law Constants

Values of between 4 and 10 are expected for the power-law creep stress exponent, *n*. Typically one determines values for *n* by fitting the model to physical creep data.

Creep data [172] collected for samples of average grain size 250µm, tested at 850°C for a range of stresses, was used to calculate the power-law creep exponent, $n = 8$, and the dimensionless constant, $A = 7x10^{12}$. The data was considered appropriate for three reasons.

1. At a grain size of 250µm it is assumed that the strain contribution associated with diffusional flow is much less than that of power-law creep.
2. Temperature is sufficiently high to assume both glide and climb of dislocations is occurring.
3. Applied stress is large enough for power-law creep to dominate the deformation process.

A.3 Dislocation Core Diffusivity

There is limited research conducted on the measurement of the dislocation core diffusivities of Fe-Ni-Cr austenitic stainless steels. Because the creep performance at low stresses and high temperatures was of primary concern in this study, it was not considered critical to find precise values. Frost and Ashby

documented core diffusivity values for fcc iron and these will be used in the current model. Equation 8.4 describes core diffusion

$$a_c D_C = a_c D_{C0}\, exp\left(-\frac{Q_C}{RT}\right)$$

Equation 8.4

with

$a_c D_{C0}$ = 1 x 10^{-23} m^4 s^{-1}	pre-exponential factor for dislocation core diffusivity
Q_C = 174 x 10^3 J mol^{-1}	activation energy for dislocation core diffusion

A.4 Modelling Diffusional Flow

Diffusional flow dominates at higher temperatures and lower stresses. When both lattice and grain boundary diffusion are considered, the rate equation for diffusional flow, $\dot{\gamma_D}$, is

$$\dot{\gamma_D} = \frac{42\ \sigma_s\ \Omega}{k\ T\ \bar{d}^2}\ D_{eff,D}$$

Equation 8.5

where $D_{eff,D}$ is the effective diffusivity from both lattice and grain boundary diffusion

$$D_{eff,D} = D_v\left[1 + \frac{\pi\delta}{\bar{d}}\frac{D_b}{D_v}\right]$$

Equation 8.6

where $\bar{d}$ is average grain size [m], atomic volume, Ω, is 1.21 x $10^{-29}m^3$, grain boundary thickness, δ, is 5 x10^{-10}m and all other parameter are as described above.

A.5 Lattice Diffusion

Lattice diffusion refers to the thermally activated movement of atoms through grain volumes. In power-law creep, lattice diffusion aids the movement of dislocations around grain volumes; while in diffusional creep, lattice diffusion is the mechanism by which atoms migrate from the boundaries experiencing

compressive stresses to those that are in tension. The diffusivity of element '*A*' through the lattice is given by

$$D_V^A = D_{V0}^A \exp\left(-\frac{Q_V^A}{RT}\right)$$

Equation 8.7

where

D_V^A	the lattice diffusivity of element A [m^2s^{-1}]
D_{V0}^A	the pre-exponential factor for lattice diffusivity of element A [m^2s^{-1}]
Q_V^A	activation energy for lattice diffusivity of element A [J mol^{-1}]
T	temperature at which diffusion is occurring [K]
R = 8.314 J mol^{-1} K^{-1}	Universal Gas Constant

Pre-exponential factors and activation energies for the lattice diffusivities for Fe, Ni, and Cr, in alloy 800 are given in Table A**.8.1**.

Table A.8.1 Pre-exponential factors and activation energies for the lattice diffusivities for Fe, Ni, and Cr, in Alloy 800.

Element	Symbol	Value	Reference
Iron (Fe)	D_{V0}^{Fe}	3.26 x 10^{-5} m^2s^{-1}	[149]
	Q_V^{Fe}	259.6 x 10^3 J mol^{-1}	
Nickel (Ni)	D_{V0}^{Ni}	8.62 x 10^{-5} m^2 s^{-1}	[173]
	Q_V^{Ni}	255.9 x 10^3 J mol^{-1}	
Chromium (Cr)	D_{V0}^{Cr}	3.24 x 10^{-4} m^2 s^{-1}	[174]
	Q_V^{Cr}	287.4 x 10^3 J mol^{-1}	

The overall lattice diffusivity for Alloy 800H is calculated from the atomic fractions, *N*, weighted mean of the diffusivities for Fe, Ni, and Cr.

$$D_v = N^{Fe} D_v^{Fe} + N^{Ni} D_v^{Ni} + N^{Cr} D_v^{Cr}$$

Equation 8.8

A.6 Boundary Diffusion

Grain boundary diffusion is responsible for the deformation associated with Coble creep. Due to the relatively open nature of grain boundaries, it is expected that diffusion along these pathways will be significantly faster than that observed in lattice diffusion. The equation for the diffusivity of element *'A'* along the grain boundaries is

$$D_B^A = D_{B0}^A \exp\left(-\frac{Q_B^A}{RT}\right)$$

Equation 8.9

where

D_B^A	the grain boundary diffusivity of element A [m^2s^{-1}]
D_{B0}^A	the pre-exponential factor for grain boundary diffusivity of element A [m^2s^{-1}]
Q_B^A	activation energy for grain boundary diffusivity of element A [$Jmol^{-1}$]
T	temperature [K]
R = 8.314 J $mol^{-1}K^{-1}$	Universal Gas Constant

Pre-exponential factors and activation energies for the boundary diffusivities for Fe, Ni, and Cr, in Alloy 800 are given in Table A.8.2.

Table A.8.2 Pre-exponential factors and activation energies for the boundary diffusivities for Fe, Ni, and Cr, in Alloy 800.

Element	Symbol	Value	Reference
Iron (Fe)	D_{B0}^{Fe}	1.88 x 10^{-5} m^2 s^{-1}	[149]
	Q_B^{Fe}	160.7 x 10^3 J mol^{-1}	
Nickel (Ni)	D_{B0}^{Ni}	3.82 x 10^{-5} m^2 s^{-1}	[173]
	Q_B^{Ni}	156.4 x 10^3 J mol^{-1}	
Chromium (Cr)	D_{B0}^{Cr}	5.80 x 10^{-5} m^2 s^{-1}	[174]
	Q_B^{Cr}	184.2 x 10^3 J mol^{-1}	

The overall boundary diffusivity for Alloy 800H is calculated from the atomic fractions, *N*, weighted mean of the diffusivities for Fe, Ni, and Cr.

$$D_B = N^{Fe} D_B^{Fe} + N^{Ni} D_B^{Ni} + N^{Cr} D_B^{Cr}$$

Equation 8.10

A.7 Combined Creep Model for Alloy 800H

The total shear strain rate, $\dot{\gamma}_{Total}$, is the sum of the contributions from the two independent processes occurring in parallel. The shear strain rate is a function of applied stress, σ_s, temperature, T, and average grain size, $\bar{d}$.

$$\dot{\gamma}_{Total}\left(\sigma_s, T, \bar{d}\right) = \dot{\gamma}_{PL} + \dot{\gamma}_D$$

Equation 8.11

Where $\dot{\gamma}_{PL}$ is given by Equation A and $\dot{\gamma}_D$ is given by Equation 8.5.

Frost and Ashby [11] describes a threshold stress, τ_{tr}, required to overcome the presence of secondary phases and drive diffusional and dislocation creep processes. Frost and Ashby show that precipitation strengthened materials typically show threshold stress equal to approximately $10^{-4}\mu$, where μ is the shear modulus. Employing the threshold stress, τ_{tr}, σ_s in Equation 8.11 is replaced with the term $\sigma_s - \tau_{tr}$.

A.8 Modelling the Effect of Grain Size Distribution on Creep Rate

Equation 8.11 is modified by replacing the average grain size, $\bar{d}$, with the probability density function (pdf) for a lognormal distribution

$$f(d|M,D) = \frac{1}{dD\sqrt{2\pi}} e^{\frac{-(\ln d - M)^2}{2D^2}}$$

Equation 8.12

where D describes the spread of the grain size distribution and is given by

$$D = \sqrt{\ln\left(1 + \frac{S^2}{\bar{d}^2}\right)}$$

Equation 8.13

and M is

$$M = \ln(\bar{d}) - \frac{1}{2}D^2$$

Equation 8.14

The steady state creep rate, Equation 8.11, for the *i*th grain, $\dot{\gamma}(d_i)$, is

$$\dot{\gamma}(d_i) = \frac{A\, D_{eff,PL}\mu b}{k\, T}\left(\frac{(\sigma_s - \tau_{tr})}{\mu}\right)^n + \frac{42(\sigma_s - \tau_{tr})\,\Omega}{k\, T}\, D_v\left[\frac{1}{{d_i}^2} + \frac{\pi\delta D_b}{D_v}\frac{1}{{d_i}^3}\right]$$

Equation 8.15

where d_i is the size of the *i*th grain.

Integrating over grains of all sizes gives the total steady-state creep rate

$$\dot{\gamma}_{Total} = \int \dot{\gamma}(d_i)\, dV_i$$

Equation 8.16

where dV_i is the volume fraction associated with the grain of size d_i given by

$$dV_i = \frac{d_i^3 \, f(d_i|M,D) \, dd_i}{\int_0^\infty d_i^3 \, f(d_i|M,D) \, dd_i}$$

Equation 8.17

Using the relation [175]

$$\int_0^\infty {d_i}^q \, f(d_i|M,D) \, dd_i = e^{(qM+\frac{q^2}{2}D^2)}$$

Equation 8.18

allows Equation 8.17 to be solved

$$dV_i = \frac{d_i^3 \, f(d_i|M,D) \, dd_i}{e^{3\left(M+\frac{3}{2}D^2\right)}}$$

Equation 8.19

Substituting Equation 8.19 into Equation 8.16 gives

$$\dot{\gamma}_{Total} = \frac{A \, D_{eff,PL} \mu b}{k \, T}\left(\frac{(\sigma_s - \tau_{\mathrm{tr}})}{\mu}\right)^n \int_0^\infty \frac{d_i^3 \, f(d_i|M,D) \, dd_i}{e^{3\left(M+\frac{3}{2}D^2\right)}} + \frac{42 \, (\sigma_s - \tau_{\mathrm{tr}}) \, \Omega}{k \, T} \, D_v \left[\int_0^\infty \frac{\frac{1}{{d_i}^2} d_i^3 \, f(d_i|M,D) \, dd_i}{e^{3\left(M+\frac{3}{2}D^2\right)}} + \frac{\pi\delta D_b}{D_v} \int_0^\infty \frac{\frac{1}{{d_i}^3} d_i^3 \, f(d_i|M,D) \, dd_i}{e^{3\left(M+\frac{3}{2}D^2\right)}}\right]$$

Equation 8.20

Solve using Equation 8.18

$$\dot{\gamma}_{Total} = \frac{A \, D_{eff,PL} \mu b}{k \, T}\left(\frac{(\sigma_s - \tau_{\mathrm{tr}})}{\mu}\right)^n + \frac{42 \, (\sigma_s - \tau_{\mathrm{tr}}) \, \Omega}{k \, T} \, D_v \left[\frac{1}{e^{2\left(M+\frac{1}{2}D^2\right)}} + \frac{\pi\delta D_b}{D_v} \frac{1}{e^{3(M+D^2)}}\right]$$

Equation 8.21

Substituting Equation 8.14 into Equation 8.21 gives $\dot{\gamma}_{Total}$ in terms of average grain size, $\bar{d}$, and D,

$$\dot{\gamma}_{Total} = \frac{A \, D_{eff,PL} \mu b}{k \, T}\left(\frac{(\sigma_s - \tau_{\mathrm{tr}})}{\mu}\right)^n + \frac{42(\sigma_s - \tau_{\mathrm{tr}}) \, \Omega}{k \, T} \, D_v \left[\frac{1}{\left(\bar{d}e^{0.5D^2}\right)^2} + \frac{\pi\delta D_b}{D_v} \frac{1}{\left(\bar{d}e^{D^2}\right)^3}\right]$$

Equation 8.22

Appendix B Matlab Algorithm

```
function EBSD(samplename,numscans,tol)
% Using euler angle data imported from HKL EBSD software, pixellated
% EBSD boundaries are reconstructed as straight line element for the purpose
% of trace analysis and the identification of coherent twins.
%
%       samplename      name of the sample entered as a string
%       numscans        number of scans (EBSD maps)
%       tol             tolerance for the reconstructions

% Inputs
%       - EBSP information including position and crystal orientation
% Outputs
%       - EBSD map (B\W)
%       - Reconstructed Map (B\W)
%       - Reconstructed Map with sig3 elements in red,coherent twins in blue,
%         sig9 in green , sig27 in cyan, and all other HGB in black
%       - Average Grain Size and Grain Size Distribution using equivalent
%         circle diameter (ECD) and Saltykoc Stereographic correction
%       - Grain Boundary Distributions (Number and Length) corrected with
%         stereographic correction

mkdir('.\Results',samplename)   %Create Directory for Results

Q(1,1:4) = [1 0 0 0];
Q(2,1:4) = [0 1 0 0];
Q(3,1:4) = [0 0 1 0];
Q(4,1:4) = [0 0 0 1];
Q(5,1:4) = [0.5 0.5 0.5 0.5];
Q(6,1:4) = [0.5 -0.5 -0.5 -0.5];
Q(7,1:4) = [0.5 0.5 -0.5 0.5];
Q(8,1:4) = [0.5 -0.5 0.5 -0.5];
Q(9,1:4) = [0.5 -0.5 0.5 0.5];
Q(10,1:4) = [0.5 0.5 -0.5 -0.5];
Q(11,1:4) = [0.5 -0.5 -0.5 0.5];
Q(12,1:4) = [0.5 0.5 0.5 -0.5];
Q(13,1:4) = [1/sqrt(2) 1/sqrt(2) 0 0];
Q(14,1:4) = [1/sqrt(2) 0 1/sqrt(2) 0];
Q(15,1:4) = [1/sqrt(2) 0 0 1/sqrt(2)];
Q(16,1:4) = [1/sqrt(2) -1/sqrt(2) 0 0];
Q(17,1:4) = [1/sqrt(2) 0 -1/sqrt(2) 0];
Q(18,1:4) = [1/sqrt(2) 0 0 -1/sqrt(2)];
Q(19,1:4) = [0 1/sqrt(2) 1/sqrt(2) 0];
Q(20,1:4) = [0 -1/sqrt(2) 1/sqrt(2) 0];
Q(21,1:4) = [0 0 1/sqrt(2) 1/sqrt(2)];
Q(22,1:4) = [0 0 -1/sqrt(2) 1/sqrt(2)];
Q(23,1:4) = [0 1/sqrt(2) 0 1/sqrt(2)];
Q(24,1:4) = [0 -1/sqrt(2) 0 1/sqrt(2)];

StatData = zeros(32,numscans+1);GOSresults = 0;

for a = 1:numscans

    %Calculate the disorientation between pixels
    fprintf('Calculate disorientation between pixels...\n');
```

```
    %EBSD Data from MATLAB
    eulerraw                                                                =
dlmread(strcat('.\EBSD_Data\',samplename,'_euler_s',num2str(a),'.txt'));
    if size(eulerraw,2)>5
        eulerraw = eulerraw(:,3:7); %euler angles and raster position
    end
    res = eulerraw(2,1)-eulerraw(1,1); %Map step size
    %Map Dimensions
    max_x=(max(eulerraw(:,1))/res)+1;
    max_y=(max(eulerraw(:,2))/res)+1;
    numpix = max_x*max_y;
    %Minimum disorientation to define an HGB
    GB_thold=15;
    %Change angles to quaternions and convert data to a cell based matrix
    for b=1:(max_x*max_y)
        x=eulerraw(b,1)/res+1;
        y=eulerraw(b,2)/res+1;
        q=euler2quat(eulerraw(b,3),eulerraw(b,4),eulerraw(b,5));
        eulercell{y,x}=q;
    end
    %Calculate the disorientation between each pixel
    for y=1:max_y
        for x=1:max_x
            %Extract Quaternion for Current Pixel
            quat = eulercell{y,x};
            if x~=max_x
               %Extract Quaternion for Pixel to the Right
               quat_r = eulercell{y,x+1};
               %Calculate disorentation
               qmis = quatnormalize(quatmultiply(quatinv(quat),quat_r));
               for n=1:24
                   qmissymm = quatnormalize(quatmultiply(qmis,Q(n,1:4)));
                   qmisALL(n) = abs(acosd(2*qmissymm(1,1)^2-1));
               end
               finaldisangright(y,x) = min(qmisALL);
            end
            if y~=max_y
               %Extract Quaternion for Pixel Below
               quat_d = eulercell{y+1,x};
               %Calculate Disorientation
               qmis = quatnormalize(quatmultiply(quatinv(quat),quat_d));
               for n=1:24
                   qmissymm = quatnormalize(quatmultiply(qmis,Q(n,1:4)));
                   qmisALL(n) = abs(acosd(2*qmissymm(1,1)^2-1));
               end
               finaldisangdown(y,x) = min(qmisALL);
            end
        end
    end

    %Define Boundaries
    fprintf('Defining grain boundaries...\n');

    %Group pixels together to form grains
    grainnos=zeros(max_y,max_x);
    cgrain=0;
    %Loop through while there is still any pixel not assigned to a grain
    while nnz(grainnos)<numpix
        %Find first pixel that has not been assigned to a grain
```

```
    [yc xc]=find(grainnos==0,1);
    tocheck=[yc xc];
    %Assign it the next available grain number
    cgrain=cgrain+1;
    grainnos(yc,xc)=cgrain;
    while size(tocheck,1)>0
        yc=tocheck(1,1);
        xc=tocheck(1,2);
        %Check which neighbours are same grain
        if yc>1 && grainnos(yc-1,xc)==0
            if finaldisangdown(yc-1,xc)<GB_thold
                grainnos(yc-1,xc)=cgrain;
                if ismember([yc-1 xc],tocheck,'rows')==0
                    tocheck=[tocheck;yc-1,xc];
                end
            end
        end
        if yc<max_y && grainnos(yc+1,xc)==0
            if finaldisangdown(yc,xc)<GB_thold
                grainnos(yc+1,xc)=cgrain;
                if ismember([yc+1 xc],tocheck,'rows')==0
                    tocheck=[tocheck;yc+1,xc];
                end
            end
        end
        if xc<max_x && grainnos(yc,xc+1)==0
            if finaldisangright(yc,xc)<GB_thold
                grainnos(yc,xc+1)=cgrain;
                if ismember([yc xc+1],tocheck,'rows')==0
                    tocheck=[tocheck;yc,xc+1];
                end
            end
        end
        if xc>1 && grainnos(yc,xc-1)==0
            if finaldisangright(yc,xc-1)<GB_thold
                grainnos(yc,xc-1)=cgrain;
                if ismember([yc xc-1],tocheck,'rows')==0
                    tocheck=[tocheck;yc,xc-1];
                end
            end
        end
        %Delete the line just checked
        tocheck(1,:)=[];
    end
end

%Noise Reduction
fprintf('EBSD Map Noise Reduction...\n');

grainnos_noiseR = grainnos;
done = false;
while ~done
    done = true;
    for b = 1:max(unique(grainnos_noiseR))
        [row,col] = find(grainnos_noiseR == b);
        if isempty(row) == 1
            elseif size(row,1) <= 5
                done = false;
                for c = 1:size(row,1)
                    if row(c,1) == 1 && col(c,1) == 1
```

```
                            nearest_neighbours                                  =
[NaN,NaN,NaN,NaN,grainnos(row(c,1),col(c,1)+1),NaN,grainnos(row(c,1)+1,col(c,
1)),grainnos(row(c,1)+1,col(c,1)+1)];
                        elseif row(c,1) == 1 && col(c,1) == max_x
                            nearest_neighbours                                  =
[NaN,NaN,NaN,grainnos(row(c,1),col(c,1)-1),NaN,grainnos(row(c,1)+1,col(c,1)-
1),grainnos(row(c,1)+1,col(c,1)),NaN];
                        elseif row(c,1) == max_y && col(c,1) == 1
                            nearest_neighbours    =    [NaN,grainnos(row(c,1)-
1,col(c,1)),grainnos(row(c,1)-
1,col(c,1)+1),NaN,grainnos(row(c,1),col(c,1)+1),NaN,NaN,NaN];
                        elseif row(c,1) == max_y && col(c,1) == max_x
                            nearest_neighbours      =      [grainnos(row(c,1)-
1,col(c,1)-1),grainnos(row(c,1)-1,col(c,1)),NaN,grainnos(row(c,1),col(c,1)-
1),NaN,NaN,NaN,NaN];
                        elseif row(c,1) == 1
                            nearest_neighbours                                  =
[NaN,NaN,NaN,grainnos(row(c,1),col(c,1)-
1),grainnos(row(c,1),col(c,1)+1),grainnos(row(c,1)+1,col(c,1)-
1),grainnos(row(c,1)+1,col(c,1)),grainnos(row(c,1)+1,col(c,1)+1)];
                        elseif row(c,1) == max_y
                            nearest_neighbours      =      [grainnos(row(c,1)-
1,col(c,1)-1),grainnos(row(c,1)-1,col(c,1)),grainnos(row(c,1)-
1,col(c,1)+1),grainnos(row(c,1),col(c,1)-
1),grainnos(row(c,1),col(c,1)+1),NaN,NaN,NaN];
                        elseif col(c,1) == 1
                            nearest_neighbours    =    [NaN,grainnos(row(c,1)-
1,col(c,1)),grainnos(row(c,1)-
1,col(c,1)+1),NaN,grainnos(row(c,1),col(c,1)+1),NaN,grainnos(row(c,1)+1,col(c
,1)),grainnos(row(c,1)+1,col(c,1)+1)];
                        elseif col(c,1) == max_x
                            nearest_neighbours      =      [grainnos(row(c,1)-
1,col(c,1)-1),grainnos(row(c,1)-1,col(c,1)),NaN,grainnos(row(c,1),col(c,1)-
1),NaN,grainnos(row(c,1)+1,col(c,1)-1),grainnos(row(c,1)+1,col(c,1)),NaN];
                        else
                            nearest_neighbours      =      [grainnos(row(c,1)-
1,col(c,1)-1),grainnos(row(c,1)-1,col(c,1)),grainnos(row(c,1)-
1,col(c,1)+1),grainnos(row(c,1),col(c,1)-
1),grainnos(row(c,1),col(c,1)+1),grainnos(row(c,1)+1,col(c,1)-
1),grainnos(row(c,1)+1,col(c,1)),grainnos(row(c,1)+1,col(c,1)+1)];
                        end
                        nearest_neighbours(1,find(nearest_neighbours==b))    =
NaN;
                        grainnos_noiseR(row(c,1),col(c,1))                   =
mode(nearest_neighbours);
                        ind                                                  =
find(nearest_neighbours==mode(nearest_neighbours));
                        for d = 1:size(ind,1)
                            if ind == 1
                                toaverage(d,1:4)     =     eulercell{row(c,1)-
1,col(c,1)-1};
                            elseif ind == 2
                                toaverage(d,1:4)     =     eulercell{row(c,1)-
1,col(c,1)};
                            elseif ind == 3
                                toaverage(d,1:4)     =     eulercell{row(c,1)-
1,col(c,1)+1};
                            elseif ind == 4
                                toaverage(d,1:4)                             =
eulercell{row(c,1),col(c,1)-1};
```

```
                            elseif ind == 5
                                toaverage(d,1:4)                                =
eulercell{row(c,1),col(c,1)+1};
                            elseif ind == 6
                                toaverage(d,1:4)                                =
eulercell{row(c,1)+1,col(c,1)-1};
                            elseif ind == 7
                                toaverage(d,1:4)                                =
eulercell{row(c,1)+1,col(c,1)};
                            elseif ind == 8
                                toaverage(d,1:4)                                =
eulercell{row(c,1)+1,col(c,1)+1};
                            end
                        end
                        if exist('toaverage','var')
                            eulercell{row(c,1),col(c,1)} = avquat(toaverage);
                        end
                    end
                    grainnos = grainnos_noiseR;
            end
        end
    end
    grainnos = grainnos_noiseR;

save(strcat('.\Results\',samplename,'\',samplename,'_grain_nos_s',num2str(a),
'.mat'),'grainnos');

    %Increase map size for better resolution
    [rows cols] = size(grainnos);
    grainnos_big = ones(rows*2, cols*2);
    for i = 1:rows
       for j = 1:cols
           number = grainnos(i,j);
           grainnos_big((i*2-1):i*2,(j*2-1):j*2) = number;
       end
    end

    %Create Boundary Location Matrix
    fprintf('Create Boundary Location Matrix...\n');

    [rows cols] = size(grainnos_big);
    %Create Boundary Location Cell Matrix - bLoc - contains the boundary and
the
    %grains that the boundary belong to
    EBSDmap = ones(rows,cols);
    bList = 0; %List of all pixels that represent a boundary location
    tList = 0; %List of all pixels that represent a boundary intersection
(triple point)
    for i = 1:rows
        for j = 1:cols
            if i == rows && j ~= cols
               pixID = unique([grainnos_big(i,j),grainnos_big(i,j+1)]);
            elseif j == cols && i ~= rows
               pixID = unique([grainnos_big(i,j),grainnos_big(i+1,j)]);
            elseif i == rows && j == cols
               pixID = unique([grainnos_big(i,j)]);
            else
               pixID                                                            =
unique([grainnos_big(i,j),grainnos_big(i+1,j),grainnos_big(i,j+1),grainnos_bi
g(i+1,j+1)]);
```

```
            end
            bLoc{i,j} = pixID;
            if size(pixID,2) == 2
               bList(end+1,1:size(pixID,2)) = pixID;
               bList(end,3:4) = [i,j];
               EBSDmap(i,j) = 0;
            elseif size(pixID,2) > 2
                tList(end+1,1:size(pixID,2)) = pixID;
                tList(end,5:6) = [i,j];
                EBSDmap(i,j) = 0;
            end
            if (i == 1 || i == rows) && size(pixID,2) == 2
                tList(end+1,1:size(pixID,2)) = pixID;
                tList(end,5:6) = [i,j];
                EBSDmap(i,j) = 0;
            end
            if (j == 1 || j == cols) && size(pixID,2) == 2
                tList(end+1,1:size(pixID,2)) = pixID;
                tList(end,5:6) = [i,j];
                EBSDmap(i,j) = 0;
            end
        end
    end
    bList(1,:) = [];tList(1,:) = [];
    uniquebList      =      sortrows(unique(bList(:,1:2),'rows'));tList      =
sortrows(tList);

imwrite(EBSDmap,strcat('.\Results\',samplename,'\',samplename,'_EBSD_s',num2s
tr(a),'.png'));

    %Recrystallization Fraction
    fprintf('Calculate Recrystallization Fraction...\n');


[recrymap,recryfraction,resultf]=GOS(grainnos,eulercell,EBSDmap,grainnos_big)
;
    GOSresults(end+1:end+length(resultf),1:2)=resultf;

imwrite(recrymap,strcat('.\Results\',samplename,'\',samplename,'_recrystalliz
ation_s',num2str(a),'.png'));

    %Reconstructing Boundaries
    fprintf('Reconstructing Boundaries...\n');

    results_complete = zeros(1,22);
    %Reconstruct EBSD traces with straight line segments
    grain_counter = 0;
    for b = 1:size(uniquebList,1)
        bCoord = 0; %list of coordinates for the boundary of interest
        [a1 b1]=find(bList(:,1:2)==uniquebList(b,1));
        [c1 d1]=find(bList(:,1:2)==uniquebList(b,2));
        rows=intersect(a1,c1);
        for c = 1:length(rows)
            bCoord(end+1,1:2) = [bList(rows(c),3),bList(rows(c),4)];
        end
        bCoord(1,:)=[];
        tCoord = 0; %list of triple point coordinates
        [a1 b1]=find(tList(:,1:4)==uniquebList(b,1));
        [c1 d1]=find(tList(:,1:4)==uniquebList(b,2));
        rows=intersect(a1,c1);
```

```
        %Grab the co-ordinates
        for c = 1:length(rows)
            bCoord(end+1,1:2) = [tList(rows(c),5),tList(rows(c),6)];
            tCoord(end+1,1:2) = [tList(rows(c),5),tList(rows(c),6)];
        end
        tCoord(1,:)=[];

        if isempty(tCoord)==1
            tCoord(1,1:2)=bCoord(1,1:2);
        end
        bCoord = unique(bCoord,'rows');
        tCoord = unique(tCoord,'rows');
        done1= false;
        while ~done1          %Organise the pixels in order
            %traceIdx - contains the ordered indices of the point on the
trace
            traceIdx = NaN(length(bCoord),1);
            %matrix of all distances between all points
            distances                                                    =
squareform(pdist(bCoord));distances(logical(eye(length(bCoord)))) = NaN;
            %Assign a triple point for the starting traceIdx
            startIdx                                                     =
find(ismember(bCoord,tCoord(1,1:2),'rows'));traceIdx(1) = startIdx;
            %starting from startIdx: find the closest next point, store in
traceIdx,
            %check whether we've arrived at the a triple point, and repeat if
we haven't
            done = false;
            traceCt = 1;
            while ~done
                %If distance is greater than 1 that we are at a false
                %triple point and we have to ignore that tpoint
                if min(distances(traceIdx(traceCt),:))>1
                    done = true;
                end
                %find the index of the next, closest point - newIdx
                [~,newIdx] = min(distances(traceIdx(traceCt),:));
                %store new index
                traceCt = traceCt + 1;
                traceIdx(traceCt) = newIdx;
                %check whether we're done
                [index] = ismember(tCoord,bCoord(newIdx,1:2),'rows');
                if isempty(find(index))==0
                    endIdx                                               =
find(ismember(bCoord,tCoord(find(index),1:2),'rows'));
                    done = true;
                else
                    %# mask the backward distance so that there's no turning
back
                    distances(newIdx,traceIdx(traceCt-1)) = NaN;
                end
            end
            if min(distances(traceIdx(traceCt-1),:))>1
                tCoord(1,:)=[];
                if isempty(tCoord)==1||size(tCoord,1)==1;
                   done1=true;
                end
            else
                traceIdx(~isfinite(traceIdx)) = [];
                pos=bCoord(traceIdx,1:2);
```

```
                bCoord(traceIdx,:)=[];
                if isempty(bCoord)==1
                    done = true;
                end
                pos1 = fliplr(pos);
                pos1 = [pos1(:,1),abs(pos1(:,2)-max_x*2)];
                %Use Douglas Peucker Polyline Simplification
                [ps ix] = dpsimplify(pos1,2*tol);
                ps = fliplr(ps);
                ps = [abs(ps(:,1)-max_y*2),ps(:,2)];
                %Remove original nodes from tCoord
                tCoord(find(ismember(tCoord,pos,'rows')),:)=[];
                if isempty(tCoord)==1||size(tCoord,1)==1;
                   done1=true;
                end
                if size(ps,1)==1
                    ps=pos1;
                end
                grain_counter = grain_counter + 1;
                results_complete(end+1,1) = grain_counter; %Boundary Number
in Results
                results_complete(end,2:3) = uniquebList(b,1:2); %The numbers
of the grains that form the boundary
                %Calculates the average orientation of the grains at the
boundaries
                grainnos_dialate = grainnos_big;
                eulercell_check = eulercell;
                %zero out all the pixels near the boundary in
grainnos_dialate so
                %they can't be selected for averaging
                for c = 1:size(pos,1)
                    if
pos(c,1)==1||pos(c,1)==2||pos(c,1)==max_y*2||pos(c,1)==max_y*2-
1||pos(c,2)==1||pos(c,2)==2||pos(c,2)==max_x*2||pos(c,2)==max_x*2-1
                    else
                        mask = zeros(5,5);
                        grainnos_dialate(pos(c,1)-2:pos(c,1)+2,pos(c,2)-
2:pos(c,2)+2)=mask;
                    end
                end
                %for each grain find that average orienation at the current
                %boundary
                for cc = 1:2
                    [r,c]=find(grainnos_dialate==results_complete(end,1+cc));
                    grainPix = [r,c];

grainPix(any(grainPix'==1),:)=[];grainPix(grainPix(:,1)==max_y*2,:)=[];grainP
ix(grainPix(:,2)==max_x*2,:)=[];
                    quats_average=0;
                    if isempty(grainPix)==1
                        %select all the angles in the grains to average -
used for
                        %very small grains where it is not possible to select
                        %angles more than 1 pixel away from the boundary
                        [r,c]=find(grainnos==results_complete(end,1+cc));
                        grainPix2 = [r,c];
                        quats_average=0;
                        for d=1:size(grainPix2,1)
```

```
quats_average(end+1,1:4)=eulercell{ceil(grainPix2(d,1)/2),ceil(grainPix2(d,2)
/2)};
                        end
                        quats_average(1,:)=[];

results_complete(end,cc*4:cc*4+3)=avquat(quats_average);
                    else
                        for d = 1:size(grainPix,1)
                            if   isempty(find([grainnos_dialate(grainPix(d,1)-
1,grainPix(d,2)-1),grainnos_dialate(grainPix(d,1)-
1,grainPix(d,2)),grainnos_dialate(grainPix(d,1)-
1,grainPix(d,2)+1),grainnos_dialate(grainPix(d,1),grainPix(d,2)-
1),grainnos_dialate(grainPix(d,1),grainPix(d,2)+1),grainnos_dialate(grainPix(
d,1)+1,grainPix(d,2)-
1),grainnos_dialate(grainPix(d,1)+1,grainPix(d,2)),grainnos_dialate(grainPix(
d,1)+1,grainPix(d,2)+1)]==0))==0                                                &
eulercell_check{ceil(grainPix(d,1)/2),ceil(grainPix(d,2)/2)}~=0

quats_average(end+1,1:4)=eulercell{ceil(grainPix(d,1)/2),ceil(grainPix(d,2)/2
)};

eulercell_check{ceil(grainPix(d,1)/2),ceil(grainPix(d,2)/2)}=0;
                            end
                        end
                        quats_average(1,:)=[];
                        if isempty(quats_average)==1
                            [r,c]=find(grainnos==results_complete(end,1+cc));
                            grainPix2 = [r,c];
                            quats_average=0;
                            for d=1:size(grainPix2,1)

quats_average(end+1,1:4)=eulercell{ceil(grainPix2(d,1)/2),ceil(grainPix2(d,2)
/2)};
                            end
                            quats_average(1,:)=[];
                        end

results_complete(end,cc*4:cc*4+3)=avquat(quats_average);
                    end
                end
                results_complete(end:end+(size(ps,1)-2),12:15)              =
[ps(1:(size(ps,1)-1),1:2),ps(2:end,1:2)];    %The coordinates of the line
segments
                %Loop calculates segment length and the angle the segment
makes
                %with the horizontal axis
                for c = 1:size(ps,1)-1
                    seg_nodes             =             [results_complete(end-
(size(ps,1))+(c+1),12:13);results_complete(end-(size(ps,1))+(c+1),14:15)];
                    results_complete(end-(size(ps,1))+(c+1),16)             =
pdist(seg_nodes)/2*res;
                    seg_nodes = sortrows(seg_nodes,1);
                    v1     =     [seg_nodes(1,2)-seg_nodes(2,2),seg_nodes(2,1)-
seg_nodes(1,1)];
                    v2 = [1,0];
                    results_complete(end-(size(ps,1))+(c+1),17)             =
acosd(dot(v1,v2)/(norm(v1)*norm(v2)));
                end
                %Calculate Boundary Type
```

```
                results_complete(end-size(ps,1)+2:end,18:22)                =
boundarytype([results_complete((end+2-
size(ps,1)):end,4:11),results_complete((end+2-size(ps,1):end),17)]);
            end
        end
    end
    results_complete(1,:)=[];

xlswrite(strcat('.\Results\',samplename,'\',samplename,'_boundaryRecon.xls'),
results_complete,a,'A2');

xlswrite(strcat('.\Results\',samplename,'\',samplename,'_boundaryRecon.xls'),
{'boundary        number','grainID        1','grainID        2','quaternion
1','','','','quaternion  2','','','','row  start','col  start','row  end','col
end','length','angle','CSL','Deviation','grain     1     angle','grain     2
angle','Twin=1'},a,'A1');

    %Calculate Grain Size and Stats
    grainsizes_notwins = 0;
    %Calculate 2D grain size - Coherent Twins Removed
    twinDefine = 0.5; %Length of a sig3 boundary that needs to be coherent
for it to be considered a coherent twin boundary
    boundaries = unique(results_complete(:,1)); %boundaries that exist in the
map
    boundaries(find(boundaries==0)) = []; %remove the 0 from the list
    twin_boundaries = zeros(1,3);
    for b = 1:size(boundaries,1)
        row = find(results_complete(:,1)==boundaries(b,1));
        if  results_complete(row,18)  ==  3;  %the  boundary  is  a  sig3  and
potential twin
            %Look at all the rows associated with that boundary and tally
            %the total length and the length considered coherent
            row_counter = 0;length_total = 0;length_twin = 0;
            done = false;
            while ~done
                length_total            =            length_total            +
results_complete(row+row_counter,16);
                if results_complete(row+row_counter,22) == 1;
                    length_twin           =           length_twin           +
results_complete(row+row_counter,16);
                end
                row_counter = row_counter + 1;
                if results_complete(row+row_counter) ~= 0
                    done = true;
                end
            end
            if length_twin/length_total >= twinDefine;
               twin_boundaries(end+1,1:3) = [results_complete(row,2:3),1];
            end
        end
    end
    twin_boundaries(1,:) = [];
    twin_boundaries = unique(twin_boundaries,'rows');
    grainnos_big_notwins = grainnos_big;
    for b = 1:size(twin_boundaries,1)
        grainnos_big_notwins(grainnos_big_notwins==twin_boundaries(b,2))     =
twin_boundaries(b,1);
        twin_boundaries(twin_boundaries==twin_boundaries(b,2))               =
twin_boundaries(b,1);
    end
```

```
save(strcat('.\Results\',samplename,'\',samplename,'_grain_nos_EXCLtwins_s',num2str(a),'.mat'),'grainnos_big_notwins')
    grains_notwins = unique(grainnos_big_notwins);

    edge_grains                                                                    =
unique([unique(grainnos_big_notwins(:,1));unique(grainnos_big_notwins(:,end))
;unique(grainnos_big_notwins(1,:))';unique(grainnos_big_notwins(1,end))'])
    for b = 1:size(grains_notwins,1)
        if isempty(find(ismember(edge_grains,grains_notwins(b))))==1
            grainsizes_notwins(b,1)                                                =
sqrt(4*res^2*(length(find(grainnos_big_notwins==grains_notwins(b,1))))/4/pi);
        end
    end


xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),grainsizes_notwins,2,strcat(char(96+a),num2str(1)));
    StatData(1,a) = mean(grainsizes_notwins);
    StatData(3,a) = length(grainsizes_notwins);
    StatData(2,a)                                                                  =
sqrt(log(1+(std(grainsizes_notwins))^2/(mean(grainsizes_notwins))^2));

    %Calculate 3D grain size
    [StatData(4,a),StatData(6,a),grainsize3D] = saltykov(grainsizes_notwins);
    StatData(5,a) = sqrt(log(1+(std(grainsize3D))^2/(mean(grainsize3D))^2));

xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),grainsize3D,3,strcat(char(96+a),num2str(1)));

    %Recrystallization Fraction
    StatData(7,a) = 100*recryfraction;

    %Compile Boundary Stats
    TwinNumber = 0;sig3Number = 0;sig9Number = 0;sig27Number = 0;otherNumber
= 0;
    TwinLength = 0;sig3Length = 0;sig9Length = 0;sig27Length = 0;otherLength
= 0;

    boundaries = unique(results_complete(:,1)); %boundaries that exist in the
map
    boundaries(find(boundaries==0)) = []; %remove the 0 from the list
    for b=1:size(boundaries,1)
        row = find(results_complete(:,1)==boundaries(b,1));
        %Look at all the rows associated with that boundary
        row_counter = 0;length_total = 0;Coherent_Length=0;
        done = false;
        while ~done
            length_total              =              length_total              +
results_complete(row+row_counter,16);
            if results_complete(row+row_counter,22) == 1;
                Coherent_Length          =          Coherent_Length          +
results_complete(row+row_counter,16);
            end
            row_counter = row_counter + 1;
            if results_complete(row+row_counter) ~= 0
                done = true;
            end
        end
```

```
        if          results_complete(row,18)          ==          3          &&
(Coherent_Length/length_total)>=0.5;
            TwinNumber = TwinNumber +1;
            sig3Length = sig3Length + length_total - Coherent_Length;
            TwinLength = TwinLength + Coherent_Length;
        elseif         results_complete(row,18)         ==         3         &&
(Coherent_Length/length_total)<0.5;
            sig3Number = sig3Number +1;
            sig3Length = sig3Length + length_total - Coherent_Length;
            TwinLength = TwinLength + Coherent_Length;
        elseif results_complete(row,18) == 9;
            sig9Length = sig9Length + length_total;
            sig9Number = sig9Number +1;
        elseif results_complete(row,18) == 27;
            sig27Length = sig27Length + length_total;
            sig27Number = sig27Number +1;
        elseif results_complete(row,18) ~= 3;
            otherLength = otherLength + length_total;
            otherNumber = otherNumber +1;
        end

    end

    %Calculate Length of Different Boundary Types
    StatData(8,a) = TwinLength;
    StatData(9,a) = sig3Length;
    StatData(10,a) = sig9Length;
    StatData(11,a) = sig27Length;
    StatData(12,a) = otherLength;

    %Calculate Length Fraction of Different Boundary Types
    StatData(13,a) = 100*TwinLength/sum(StatData(8:12,a));
    StatData(14,a) = 100*sig3Length/sum(StatData(8:12,a));
    StatData(15,a) = 100*sig9Length/sum(StatData(8:12,a));
    StatData(16,a) = 100*sig27Length/sum(StatData(8:12,a));
    StatData(17,a) = 100*otherLength/sum(StatData(8:12,a));

    %Calculate Number for the Different Boundary Types
    StatData(18,a) = TwinNumber;
    StatData(19,a) = sig3Number;
    StatData(20,a) = sig9Number;
    StatData(21,a) = sig27Number;
    StatData(22,a) = otherNumber;

    %Calculate the Number Fraction for the Different Boundary Types
    StatData(23,a) = 100*TwinNumber/sum(StatData(18:22,a));
    StatData(24,a) = 100*sig3Number/sum(StatData(18:22,a));
    StatData(25,a) = 100*sig9Number/sum(StatData(18:22,a));
    StatData(26,a) = 100*sig27Number/sum(StatData(18:22,a));
    StatData(27,a) = 100*otherNumber/sum(StatData(18:22,a));

    %Calculate the Estimated 3D Number Fraction for the Different Boundary
Types
    StatData(28:32,a)                                                                =
(StatData(23:27,a)./(StatData(8:12,a)./StatData(18:22,a)))./sum(StatData(23:2
7,a)./(StatData(8:12,a)./StatData(18:22,a)))*100;
    %Account for when sig27 Number = 0
    if StatData(21,a) == 0
```

```
       StatData([28:30,32],a)                                                    =
(StatData([23:25,27],a)./(StatData([8:10,12],a)./StatData([18:20,22],a)))./su
m(StatData([23:25,27],a)./(StatData([8:10,12],a)./StatData([18:20,22],a)))*10
0;
    end
    %Account for when sig9 Number = 0
    if StatData(22,a) == 0
       StatData([28:29,31:32],a)                                                 =
(StatData([23:24,26:27],a)./(StatData([8:9,11:12],a)./StatData([18:19,21:22],
a)))./sum(StatData([23:24,26:27],a)./(StatData([8:9,11:12],a)./StatData([18:1
9,21:22],a)))*100;
    end

    %EBSD Map - Coherent Twins Excluded
    EBSD_Map_NoTwins = ones(size(EBSDmap,1),size(EBSDmap,2));
    for y = 1:size(EBSD_Map_NoTwins,1)
        for x = 1:size(EBSD_Map_NoTwins,2)
            if  x~=size(EBSD_Map_NoTwins,2)  &&  grainnos_big_notwins(y,x)  ~=
grainnos_big_notwins(y,x+1)
                EBSD_Map_NoTwins(y,x) = 0;
            end
            if  y~=size(EBSD_Map_NoTwins,1)  &&  grainnos_big_notwins(y,x)  ~=
grainnos_big_notwins(y+1,x)
                EBSD_Map_NoTwins(y,x) = 0;
            end
        end
    end
    %Fill in boundary gaps
    for y = 2:size(EBSD_Map_NoTwins,1)-2
        for x = 2:size(EBSD_Map_NoTwins,2)-2
            if EBSD_Map_NoTwins(y,x) == 0 && EBSD_Map_NoTwins(y-1,x+1) == 0
&& EBSD_Map_NoTwins(y-1,x+2) == 0
               EBSD_Map_NoTwins(y-1,x) = 0;
            end
        end
    end
    for x = 1:size(EBSD_Map_NoTwins,2)
        if EBSD_Map_NoTwins(2,x) == 0
           EBSD_Map_NoTwins(1,x) = 0;
        end
    end
    for y = 1:size(EBSD_Map_NoTwins,1)
        if EBSD_Map_NoTwins(y,cols-1) == 0
           EBSD_Map_NoTwins(y,cols) = 0;
        end
    end

imwrite(EBSD_Map_NoTwins,strcat('.\Results\',samplename,'\',samplename,'_EBSD
_Twins_Removed_s',num2str(a),'.png'));

    %Reconstructed Boundary Map and Coloured Reconstructed Boundary Map -
    %Blue(coherent Twins), Red(sig3),Green(sig9), pink(sig27)

    colR = ones(size(EBSDmap,1),size(EBSDmap,2));
    colG = ones(size(EBSDmap,1),size(EBSDmap,2));
    colB = ones(size(EBSDmap,1),size(EBSDmap,2));
    mapBW = ones(size(EBSDmap,1),size(EBSDmap,2));
```

```
    [ind                                    label]                                    =
drawline(results_complete(:,12:13),results_complete(:,14:15),[size(mapBW,1)
size(mapBW,2)]);
    mapBW(ind) = 0;

imwrite(mapBW,strcat('.\Results\',samplename,'\',samplename,'_BWrecon_s',num2
str(a),'.png'));

    boundaries = unique(results_complete(:,1)); %boundaries that exist in the
map
    boundaries(find(boundaries==0)) = []; %remove the 0 from the list

    for b = 1:size(boundaries,1)
        row = find(results_complete(:,1)==boundaries(b,1));
        %Look at all the rows associated with that boundary
        row_counter = 0;length_total = 0;length_twin = 0;
        done = false;
        while ~done
            length_total              =              length_total              +
results_complete(row+row_counter,16);
            if results_complete(row+row_counter,22) == 1;
                length_twin            =             length_twin            +
results_complete(row+row_counter,16);
            end
            row_counter = row_counter + 1;
            if results_complete(row+row_counter) ~= 0
                done = true;
            end
        end
        [ind    label]    =    drawline(results_complete(row:row+row_counter-
1,12:13),results_complete(row:row+row_counter-1,14:15),[size(colR,1)
size(colR,2)]);
        if results_complete(row,18) == 9;
            colR(ind) = 0;colG(ind) = 1;colB(ind) = 0;
        elseif results_complete(row,18) == 27;
            colR(ind) = 1;colG(ind) = 0;colB(ind) = 1;
        elseif results_complete(row,18) ~= 3;
            colR(ind) = 0;colG(ind) = 0;colB(ind) = 0;
        elseif length_twin/length_total >= twinDefine;
            colR(ind) = 0;colG(ind) = 0;colB(ind) = 1;
        else
            colR(ind) = 1;colG(ind) = 0;colB(ind) = 0;
        end
    end
    colMap(:,:,1)=colR;
    colMap(:,:,2)=colG;
    colMap(:,:,3)=colB;

imwrite(colMap,strcat('.\Results\',samplename,'\',samplename,'_COLOURrecon_s'
,num2str(a),'.png'));

end

GOSresults(1,:)=[];
GOSresults=[GOSresults(:,1),GOSresults(:,2)./numscans];
xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),GOS
results,4,'A1');
```

```
xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),Sta
tData,1,'B2');
xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),{'2
D Grain Size';'2D Standard Deviation';'2D Number of Grains';'3D Grain
Size';'3D Standard Deviation';'3D Number of Grains';'%
Recrytallized';'Coherent Twin Length';'sig3 Length (ex coherent twins)';'sig9
Length';'sig27 Length';'RHGB Length';'Coherent Twin Length Fraction';'sig3
Length Fraction (ex coherent twins)';'sig9 Length Fraction';'sig27 Length
Fraction';'RHGB Length Fraction';'Coherent Twin Number';'sig3 Number (ex
coherent twins)';'sig9 Number';'sig27 Number';'RHGB Number';'Coherent Twin
Number Fraction';'sig3 Number Fraction (ex coherent twins)';'sig9 Number
Fraction';'sig27 Number Fraction';'RHGB Number Fraction';'Est3D Coherent Twin
Number Fraction';'Est3D sig3 Number Fraction (ex coherent twins)';'Est3D sig9
Number Fraction';'Est3D sig27 Number Fraction';'Est 3D RHGB Number
Fraction'},1,'A2');
for b=1:numscans

xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),{st
rcat('Scan ',num2str(b))},1,strcat(char(97+b),num2str(1)));
end
xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),{'T
otals'},1,strcat(char(97+numscans+1),num2str(1)));

grains2D                                                                =
xlsread(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),2,'A
1:AA10000');
grains2D=grains2D(:);grains2D(isnan(grains2D))=[];
grains3D                                                                =
xlsread(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),3,'A
1:AA10000');
grains3D(:);grains3D(isnan(grains3D))=[];

StatData(1,numscans+1) = mean(grains2D);
StatData(2,numscans+1) = sqrt(log(1+(std(grains2D))^2/(mean(grains2D))^2));
StatData(3,numscans+1) = length(grains2D);
StatData(4,numscans+1) = mean(grains3D);
StatData(5,numscans+1) = sqrt(log(1+(std(grains3D))^2/(mean(grains3D))^2));
StatData(6,numscans+1) = length(grains3D);
StatData(7,numscans+1) = sum(StatData(7,1:numscans))/numscans;
StatData(8:12,numscans+1) = sum(StatData(8:12,1:numscans),2);
StatData(13:17,numscans+1)                                              =
StatData(8:12,numscans+1)./sum(StatData(8:12,numscans+1))*100;
StatData(18:22,numscans+1) = sum(StatData(18:22,1:numscans),2);
StatData(23:27,numscans+1)                                              =
StatData(18:22,numscans+1)./sum(StatData(18:22,numscans+1))*100;
StatData(28:32,numscans+1)                                              =
(StatData(23:27,numscans+1)./(StatData(8:12,numscans+1)./StatData(18:22,numsc
ans+1)))./sum(StatData(23:27,numscans+1)./(StatData(8:12,numscans+1)./StatDat
a(18:22,numscans+1)))*100;
%Account for when sig27 Number = 0
if StatData(21,numscans+1) == 0
   StatData([28:30,32],numscans+1)                                      =
(StatData([23:25,27],numscans+1)./(StatData([8:10,12],numscans+1)./StatData([
18:20,22],numscans+1)))./sum(StatData([23:25,27],numscans+1)./(StatData([8:10
,12],numscans+1)./StatData([18:20,22],numscans+1)))*100;
end
%Account for when sig9 Number = 0
if StatData(22,numscans+1) == 0
   StatData([28:29,31:32],numscans+1)                                   =
(StatData([23:24,26:27],numscans+1)./(StatData([8:9,11:12],numscans+1)./StatD
```

```
ata([18:19,21:22],numscans+1))./sum(StatData([23:24,26:27],numscans+1)./(Sta
tData([8:9,11:12],numscans+1)./StatData([18:19,21:22],numscans+1)))*100;
end
xlswrite(strcat('.\Results\',samplename,'\',samplename,'_statistics.xls'),Sta
tData,1,'B2');

end
```

```
function [averagequat] = avquat(quatlist)
%Calculates the average quaternion from a list of quaternions

Q(1,1:4) = [1 0 0 0];
Q(2,1:4) = [0 1 0 0];
Q(3,1:4) = [0 0 1 0];
Q(4,1:4) = [0 0 0 1];
Q(5,1:4) = [0.5 0.5 0.5 0.5];
Q(6,1:4) = [0.5 -0.5 -0.5 -0.5];
Q(7,1:4) = [0.5 0.5 -0.5 0.5];
Q(8,1:4) = [0.5 -0.5 0.5 -0.5];
Q(9,1:4) = [0.5 -0.5 0.5 0.5];
Q(10,1:4) = [0.5 0.5 -0.5 -0.5];
Q(11,1:4) = [0.5 -0.5 -0.5 0.5];
Q(12,1:4) = [0.5 0.5 0.5 -0.5];
Q(13,1:4) = [1/sqrt(2) 1/sqrt(2) 0 0];
Q(14,1:4) = [1/sqrt(2) 0 1/sqrt(2) 0];
Q(15,1:4) = [1/sqrt(2) 0 0 1/sqrt(2)];
Q(16,1:4) = [1/sqrt(2) -1/sqrt(2) 0 0];
Q(17,1:4) = [1/sqrt(2) 0 -1/sqrt(2) 0];
Q(18,1:4) = [1/sqrt(2) 0 0 -1/sqrt(2)];
Q(19,1:4) = [0 1/sqrt(2) 1/sqrt(2) 0];
Q(20,1:4) = [0 -1/sqrt(2) 1/sqrt(2) 0];
Q(21,1:4) = [0 0 1/sqrt(2) 1/sqrt(2)];
Q(22,1:4) = [0 0 -1/sqrt(2) 1/sqrt(2)];
Q(23,1:4) = [0 1/sqrt(2) 0 1/sqrt(2)];
Q(24,1:4) = [0 -1/sqrt(2) 0 1/sqrt(2)];

for ii = 1:size(quatlist,1)
    for jj = 1:24
        qresult(jj,1:4)                                                    =
quatnormalize(quatmultiply(quatlist(ii,1:4),Q(jj,1:4)));
    end
    %Determine Initial Quaternion
    if ii==1
        temp = abs(qresult);
        row = find(temp(:,1)==max((temp(:,1))));
        QUAT(ii,1:4) = qresult(row(1,1),1:4);
    end
    %Check which quaternion has the smallest misorientation with respect
to the initial quaternion
    if ii~=1
        for kk = 1:24
            misquat                                                        =
quatnormalize(quatmultiply(quatinv(QUAT(1,1:4)),qresult(kk,1:4)));
%Misorientation Quaternion
            angle(kk,1) = abs(rad2deg(acos(2*misquat(1,1)^2-1)));
        end
         row = find(angle(:,1)==min(angle(:,1)));
        QUAT(ii,1:4) = qresult(row,1:4);
     end
end

averagequat(1,1) = mean(QUAT(:,1));
averagequat(1,2) = mean(QUAT(:,2));
averagequat(1,3) = mean(QUAT(:,3));
averagequat(1,4) = mean(QUAT(:,4));
averagequat = quatnormalize(averagequat);
```

```
function [results] = boundarytype(data)
%BOUNDARYTYPE
%Input - boundary segment angles and quaternions
%Output - Matrix with CSL value (3,9,or,27) and the sig3 that are coherent
errorONtrace = 10; %Current error on trace
results = zeros(size(data,1),5);

%Symmetry operators
Q(1,1:4) = [1 0 0 0];
Q(2,1:4) = [0 1 0 0];
Q(3,1:4) = [0 0 1 0];
Q(4,1:4) = [0 0 0 1];
Q(5,1:4) = [0.5 0.5 0.5 0.5];
Q(6,1:4) = [0.5 -0.5 -0.5 -0.5];
Q(7,1:4) = [0.5 0.5 -0.5 0.5];
Q(8,1:4) = [0.5 -0.5 0.5 -0.5];
Q(9,1:4) = [0.5 -0.5 0.5 0.5];
Q(10,1:4) = [0.5 0.5 -0.5 -0.5];
Q(11,1:4) = [0.5 -0.5 -0.5 0.5];
Q(12,1:4) = [0.5 0.5 0.5 -0.5];
Q(13,1:4) = [1/sqrt(2) 1/sqrt(2) 0 0];
Q(14,1:4) = [1/sqrt(2) 0 1/sqrt(2) 0];
Q(15,1:4) = [1/sqrt(2) 0 0 1/sqrt(2)];
Q(16,1:4) = [1/sqrt(2) -1/sqrt(2) 0 0];
Q(17,1:4) = [1/sqrt(2) 0 -1/sqrt(2) 0];
Q(18,1:4) = [1/sqrt(2) 0 0 -1/sqrt(2)];
Q(19,1:4) = [0 1/sqrt(2) 1/sqrt(2) 0];
Q(20,1:4) = [0 -1/sqrt(2) 1/sqrt(2) 0];
Q(21,1:4) = [0 0 1/sqrt(2) 1/sqrt(2)];
Q(22,1:4) = [0 0 -1/sqrt(2) 1/sqrt(2)];
Q(23,1:4) = [0 1/sqrt(2) 0 1/sqrt(2)];
Q(24,1:4) = [0 -1/sqrt(2) 0 1/sqrt(2)];

%Set up Brandon's criteria
v3thold=15/sqrt(3);
v9thold=15/sqrt(9);
v27thold=15/sqrt(27);
%Define orientation matrix for sigma 3, 9, 27a, 27b
sig3R=createM(60,1,1,1);
sig9R=createM(38.94,1,1,0);
sig27aR=createM(31.58,1,1,0);
sig27bR=createM(35.42,2,1,0);
%Calculte the misorienation across the boundary
boundarymis = quatnormalize(quatmultiply(quatinv(data(1,1:4)),data(1,5:8)));
for n=1:24
    boundarymisSYMM(n,1:4)                                                    =
quatnormalize(quatmultiply(boundarymis,Q(n,1:4)));
    angle(n,1) = abs(acosd(2*boundarymisSYMM(n,1)^2-1));
end
[disangle row] = min(angle); %This is the disorientation angle
%The disorientation axis
disaxes(1,1)=abs(boundarymisSYMM(row,2)/sqrt(1-boundarymisSYMM(row,1)^2));
disaxes(1,2)=abs(boundarymisSYMM(row,3)/sqrt(1-boundarymisSYMM(row,1)^2));
disaxes(1,3)=abs(boundarymisSYMM(row,4)/sqrt(1-boundarymisSYMM(row,1)^2));
disnorm=sort(disaxes,'descend');
Mexp=createM(disangle,abs(disnorm(1)),abs(disnorm(2)),abs(disnorm(3)));
%For sigma 3
Md=Mexp*sig3R';
v3=acosd((Md(1,1)+Md(2,2)+Md(3,3)-1)/2);
%For sigma 9
```

```
Md=Mexp*sig9R';
v9=acosd((Md(1,1)+Md(2,2)+Md(3,3)-1)/2);
%For sigma 27a
Md=Mexp*sig27aR';
v27a=acosd((Md(1,1)+Md(2,2)+Md(3,3)-1)/2);
%For sigma 27b
Md=Mexp*sig27bR';
v27b=acosd((Md(1,1)+Md(2,2)+Md(3,3)-1)/2);
if v3<v3thold
   %This is a sigma 3 boundary
   results(1,1)=3;
   results(1,2)=v3;
   %is this sig3 a coherent twin
   for a=1:size(data,1)
       %Trace Vector of boundary segment
       Trace = [cosd(data(a,9)), sind(data(a,9))];Trace = Trace./norm(Trace);
       %Search all 24 symmetry quaternions and compare to Trace for smallest
angle
       quat1 = data(1,1:4);
       for n = 1:24
           quat1SYMM = quatnormalize(quatmultiply(quat1,Q(n,1:4)));
           twin_normal     =     quatrotate(quat1SYMM,[1/sqrt(3)     1/sqrt(3)
1/sqrt(3)]);
           twin_normal = twin_normal/norm(twin_normal);
           xyplaneintercept = [twin_normal(1,2),-twin_normal(1,1)];
           xyplaneintercept = xyplaneintercept/norm(xyplaneintercept);
           anglebetween111andtrace(n,1) = acosd(dot(xyplaneintercept,Trace));
       end
       results(a,3) =  min(abs(anglebetween111andtrace(:,1)));
       %Search all 24 symmetry quaternions and compare to Trace for smallest
angle
       quat2 = data(1,5:8);
       for n = 1:24
           quat1SYMM = quatnormalize(quatmultiply(quat2,Q(n,1:4)));
           twin_normal     =     quatrotate(quat1SYMM,[1/sqrt(3)     1/sqrt(3)
1/sqrt(3)]);
           twin_normal = twin_normal/norm(twin_normal);
           xyplaneintercept = [twin_normal(1,2),-twin_normal(1,1)];
           xyplaneintercept = xyplaneintercept/norm(xyplaneintercept);
           anglebetween111andtrace(n,1) = acosd(dot(xyplaneintercept,Trace));
       end
       results(a,4) =  min(abs(anglebetween111andtrace(:,1)));
       if results(a,3)<=errorONtrace && results(a,4)<=errorONtrace
           results(a,5) = 1;
       end
   end
elseif v9<v9thold
   %This is a sigma 9 boundary
   results(1,1)=9;
   results(1,2)=v9;
elseif v27a<v27thold
   %This is a sigma 27a boundary
   results(1,1)=27;
   results(1,2)=v27a;
elseif v27b<v27thold
   %This is a sigma 27b boundary
   results(1,1)=27;
   results(1,2)=v27b;
else
   %This is none of the above
```

```
    results(1,1)=0;
    results(1,2)=0;
end

end
```

```
function [quat] = euler2quat(phi1,phi,phi2)

%Takes euler angles and returns a quaternion

    quat(1) = cosd(phi/2)*cosd((phi1+phi2)/2);
    quat(2) = sind(phi/2)*cosd((phi1-phi2)/2);
    quat(3) = sind(phi/2)*sind((phi1-phi2)/2);
    quat(4) = cosd(phi/2)*sind((phi1+phi2)/2);
    quat = quatnormalize(quat);

end
```

```
function                    [recrymap,recryfraction,resultf]                    =
GOS(grainnos,eulercell,EBSDmap,grainnos_big)
%GOS
%Takes the mapped grain numbers,grainnos,and the quaternions,eulercell.
%Finds the grain orientation spread (GOS) of each grain and calculates the
%area fraction for that grain.
%Symmetry operators
Q(1,1:4) = [1 0 0 0];
Q(2,1:4) = [0 1 0 0];
Q(3,1:4) = [0 0 1 0];
Q(4,1:4) = [0 0 0 1];
Q(5,1:4) = [0.5 0.5 0.5 0.5];
Q(6,1:4) = [0.5 -0.5 -0.5 -0.5];
Q(7,1:4) = [0.5 0.5 -0.5 0.5];
Q(8,1:4) = [0.5 -0.5 0.5 -0.5];
Q(9,1:4) = [0.5 -0.5 0.5 0.5];
Q(10,1:4) = [0.5 0.5 -0.5 -0.5];
Q(11,1:4) = [0.5 -0.5 -0.5 0.5];
Q(12,1:4) = [0.5 0.5 0.5 -0.5];
Q(13,1:4) = [1/sqrt(2) 1/sqrt(2) 0 0];
Q(14,1:4) = [1/sqrt(2) 0 1/sqrt(2) 0];
Q(15,1:4) = [1/sqrt(2) 0 0 1/sqrt(2)];
Q(16,1:4) = [1/sqrt(2) -1/sqrt(2) 0 0];
Q(17,1:4) = [1/sqrt(2) 0 -1/sqrt(2) 0];
Q(18,1:4) = [1/sqrt(2) 0 0 -1/sqrt(2)];
Q(19,1:4) = [0 1/sqrt(2) 1/sqrt(2) 0];
Q(20,1:4) = [0 -1/sqrt(2) 1/sqrt(2) 0];
Q(21,1:4) = [0 0 1/sqrt(2) 1/sqrt(2)];
Q(22,1:4) = [0 0 -1/sqrt(2) 1/sqrt(2)];
Q(23,1:4) = [0 1/sqrt(2) 0 1/sqrt(2)];
Q(24,1:4) = [0 -1/sqrt(2) 0 1/sqrt(2)];
recrymapr=zeros(size(EBSDmap,1),size(EBSDmap,2));
recrymapg=zeros(size(EBSDmap,1),size(EBSDmap,2));
recrymapb=zeros(size(EBSDmap,1),size(EBSDmap,2));
resultf=[0,0];
grain_numbers = unique(grainnos);
for a = 1:size(grain_numbers,1)
    [row,col] = find(grainnos==grain_numbers(a,1));
    resultf(end+1,2) = size(row,1)/(size(grainnos,1)*size(grainnos,2));
    if isempty(row)~=1
        toaverage=vec2mat([eulercell{find(grainnos==grain_numbers(a,1))}],4);
        grainaverage = avquat(toaverage);
        result=0;
        for e=1:size(row,1)
            qmis                                                                =
quatnormalize(quatmultiply(quatinv(grainaverage),eulercell{row(e),col(e)}));
            %Calculate misorientations for 24 symmetry operatiors
            for n=1:24
                qmissymm = quatnormalize(quatmultiply(qmis,Q(n,1:4)));
                qmisALL(n) = abs(acosd(2*qmissymm(1,1)^2-1));
            end
            result(end+1,1) = min(qmisALL);
        end
        result(1,:)=[];
    end
    resultf(end,1) = mean(result(:,1));
    if resultf(end,1)<=1
       recrymapb(find(grainnos_big==grain_numbers(a,1)))=255;
       recrymapr(find(grainnos_big==grain_numbers(a,1)))=24;
    elseif resultf(end,1)>1 && resultf(end,1)<=2
```

```
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=204;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=49;
    elseif resultf(end,1)>2 && resultf(end,1)<=3
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=178;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=73;
    elseif resultf(end,1)>3 && resultf(end,1)<=4
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=153;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=98;
    elseif resultf(end,1)>4 && resultf(end,1)<=5
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=127;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=122;
    elseif resultf(end,1)>5 && resultf(end,1)<=6
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=102;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=147;
    elseif resultf(end,1)>6 && resultf(end,1)<=7
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=76;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=171;
    elseif resultf(end,1)>7 && resultf(end,1)<=8
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=51;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=196;
    elseif resultf(end,1)>8 && resultf(end,1)<=9
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=25;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=220;
    elseif resultf(end,1)>9
        recrymapb(find(grainnos_big==grain_numbers(a,1)))=0;
        recrymapr(find(grainnos_big==grain_numbers(a,1)))=245;
    end
end
resultf(1,:)=[];
recryfraction = sum(resultf(find(resultf(:,1)<4),2));
recrymap(:,:,1)=recrymapr.*(EBSDmap);
recrymap(:,:,2)=recrymapg.*(EBSDmap);
recrymap(:,:,3)=recrymapb.*(EBSDmap);
recrymap=recrymap./255;

end
```

```
function [av_3D,numofgrains,grainsize3D] = saltykov(graindata2D)
%Takes a list of grain size diameters

%Convert diameters to areas
graindata = pi*graindata2D.^2/4;
%Select Largest area
largestgrain = max(graindata);
%Create class intervals - smallest class must be able to include smallest
%grains
classsize = 0;
done = false;
exponent = 0;
while ~done
    range = 10^-exponent*largestgrain;
    if range>min(graindata)
        classsize(end+1,1) = range;
    else
        done = true;
    end
    exponent = exponent + 0.2;
end
classsize(1)=[];
%classsizeflip = flip(sqrt(4*classsize./pi))
%Sort list into bins
for i = 1:1:size(classsize,1)-1
    classsize(i,2)                                                    =
length(find(graindata<=classsize(i,1)&graindata>classsize(i+1,1)));
end
    classsize(i+1,2) = length(find(graindata<=classsize(i+1,1)));

%Use Saltykov equation to find grains per mm3
saltyCoeff = [1.6461, -0.4561, -0.1162, -0.4149e-1, -0.1727e-1, -0.7795e-2, -
0.3684e-2,  -0.1791e-2,  -0.8841e-3,  -0.4414e-3,  -0.2218e-3,  -0.1119e-3,  -
0.5665e-4,  -0.2872e-4,  -0.1457e-4,  -0.7401e-5,  -0.3761e-5,  -0.1911e-5,  -
0.9716e-6,  -0.4939e-6,  -0.2511e-6,  -0.1277e-6,  -0.6493e-7,  -0.3302e-7,  -
0.1679e-7, -0.8537e-8, -0.4341e-8, -0.2207e-8, -0.1122e-8, -0.570e-9];
for i = 1:size(classsize,1)

    temp = 0;
    j = 0;

    while j < 30 && (i-j)>0

        temp = temp + saltyCoeff(j+1)*classsize((i-j),2);
        j = j+1;

    end
    classsize(i,3) = temp;

    if classsize(i,3)<0
        classsize(i,3)= 0;
    end
end
%randomly generate individual grain areas from uniform distribution
grainsize3D = 0;
for i = 1:size(classsize,1)-1
    threeDgrainsizedata                                               =
randi([ceil(classsize(i+1,1)),ceil(classsize(i,1))],ceil(classsize(i,3)),1);
```

```
    grainsize3D(end+1:end+length(threeDgrainsizedata),1)                    =
threeDgrainsizedata;
end
grainsize3D(1,1)=max(graindata);
grainsize3D = (sqrt(4*grainsize3D./pi));
av_3D = mean(grainsize3D);
numofgrains = length(grainsize3D);

end
```

```
function [ps,ix] = dpsimplify(p,tol)

% Recursive Douglas-Peucker Polyline Simplification, Simplify
%
% [ps,ix] = dpsimplify(p,tol)
%
% dpsimplify uses the recursive Douglas-Peucker line simplification
% algorithm to reduce the number of vertices in a piecewise linear curve
% according to a specified tolerance. The algorithm is also know as
% Iterative Endpoint Fit. It works also for polylines and polygons
% in higher dimensions.
%
% In case of nans (missing vertex coordinates) dpsimplify assumes that
% nans separate polylines. As such, dpsimplify treats each line
% separately.
%
% For additional information on the algorithm follow this link
% http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm
%
% Input arguments
%
%     p     polyline n*d matrix with n vertices in d
%           dimensions.
%     tol   tolerance (maximal euclidean distance allowed
%           between the new line and a vertex)
%
% Output arguments
%
%     ps    simplified line
%     ix    linear index of the vertices retained in p (ps = p(ix))
%
% Examples
%
% 1. Simplify line
%
%     tol    = 1;
%     x      = 1:0.1:8*pi;
%     y      = sin(x) + randn(size(x))*0.1;
%     p      = [x' y'];
%     ps     = dpsimplify(p,tol);
%
%     plot(p(:,1),p(:,2),'k')
%     hold on
%     plot(ps(:,1),ps(:,2),'r','LineWidth',2);
%     legend('original polyline','simplified')
%
% 2. Reduce polyline so that only knickpoints remain by
%    choosing a very low tolerance
%
%     p = [(1:10)' [1 2 3 2 4 6 7 8 5 2]'];
%     p2 = dpsimplify(p,eps);
%     plot(p(:,1),p(:,2),'k+--')
%     hold on
%     plot(p2(:,1),p2(:,2),'ro','MarkerSize',10);
%     legend('original line','knickpoints')
%
% 3. Simplify a 3d-curve
%
%     x = sin(1:0.01:20)';
%     y = cos(1:0.01:20)';
```

```
%     z = x.*y.*(1:0.01:20)';
%     ps = dpsimplify([x y z],0.1);
%     plot3(x,y,z);
%     hold on
%     plot3(ps(:,1),ps(:,2),ps(:,3),'k*-');
%
%
%
% Author: Wolfgang Schwanghart, 13. July, 2010.
% w.schwanghart[at]unibas.ch


if nargin == 0
    help dpsimplify
    return
end

error(nargchk(2, 2, nargin))

% error checking
if ~isscalar(tol) || tol<0;
    error('tol must be a positive scalar')
end



% nr of dimensions
nrvertices    = size(p,1);
dims    = size(p,2);

% anonymous function for starting point and end point comparision
% using a relative tolerance test
compare = @(a,b) abs(a-b)/max(abs(a),abs(b)) <= eps;

% what happens, when there are NaNs?
% NaNs divide polylines.
Inan      = any(isnan(p),2);
% any NaN at all?
Inanp     = any(Inan);

% if there is only one vertex
if nrvertices == 1 || isempty(p);
    ps = p;
    ix = 1;

% if there are two
elseif nrvertices == 2 && ~Inanp;
    % when the line has no vertices (except end and start point of the
    % line) check if the distance between both is less than the tolerance.
    % If so, return the center.
    if dims == 2;
        d    = hypot(p(1,1)-p(2,1),p(1,2)-p(2,2));
    else
        d    = sqrt(sum((p(1,:)-p(2,:)).^2));
    end

    if d <= tol;
        ps = sum(p,1)/2;
        ix = 1;
```

```
    else
        ps = p;
        ix = [1;2];
    end

elseif Inanp;

    % case: there are nans in the p array
    % --> find start and end indices of contiguous non-nan data
    Inan = ~Inan;
    sIX = strfind(Inan',[0 1])' + 1;
    eIX = strfind(Inan',[1 0])';

    if Inan(end)==true;
        eIX = [eIX;nrvertices];
    end

    if Inan(1);
        sIX = [1;sIX];
    end

    % calculate length of non-nan components
    lIX = eIX-sIX+1;
    % put each component into a single cell
    c   = mat2cell(p(Inan,:),lIX,dims);

    % now call dpsimplify again inside cellfun.
    if nargout == 2;
        [ps,ix]   = cellfun(@(x) dpsimplify(x,tol),c,'uniformoutput',false);
        ix                                      =     cellfun(@(x,six)     x+six-
1,ix,num2cell(sIX),'uniformoutput',false);
    else
        ps   = cellfun(@(x) dpsimplify(x,tol),c,'uniformoutput',false);
    end

    % write the data from a cell array back to a matrix
    ps = cellfun(@(x) [x;nan(1,dims)],ps,'uniformoutput',false);
    ps = cell2mat(ps);
    ps(end,:) = [];

    % ix wanted? write ix to a matrix, too.
    if nargout == 2;
        ix = cell2mat(ix);
    end


else


% if there are no nans than start the recursive algorithm
ixe     = size(p,1);
ixs     = 1;

% logical vector for the vertices to be retained
I   = true(ixe,1);

% call recursive function
p   = simplifyrec(p,tol,ixs,ixe);
```

```
ps  = p(I,:);

% if desired return the index of retained vertices
if nargout == 2;
    ix  = find(I);
end

end

% _____________________________________________________
function p  = simplifyrec(p,tol,ixs,ixe)

    % check if startpoint and endpoint are the same
    % better comparison needed which included a tolerance eps

    c1 = num2cell(p(ixs,:));
    c2 = num2cell(p(ixe,:));

    % same start and endpoint with tolerance
    sameSE                                                              =
all(cell2mat(cellfun(compare,c1(:),c2(:),'UniformOutput',false)));


    if sameSE;
        % calculate the shortest distance of all vertices between ixs and
        % ixe to ixs only
        if dims == 2;
            d         =  hypot(p(ixs,1)-p(ixs+1:ixe-1,1),p(ixs,2)-p(ixs+1:ixe-
1,2));
        else
            d    = sqrt(sum(bsxfun(@minus,p(ixs,:),p(ixs+1:ixe-1,:)).^2,2));
        end
    else
        % calculate shortest distance of all points to the line from ixs to
ixe
        % subtract starting point from other locations
        pt = bsxfun(@minus,p(ixs+1:ixe,:),p(ixs,:));

        % end point
        a = pt(end,:)';

        beta = (a' * pt')./(a'*a);
        b    = pt-bsxfun(@times,beta,a)';
        if dims == 2;
            % if line in 2D use the numerical more robust hypot function
            d    = hypot(b(:,1),b(:,2));
        else
            d    = sqrt(sum(b.^2,2));
        end
    end

    % identify maximum distance and get the linear index of its location
    [dmax,ixc] = max(d);
    ixc  = ixs + ixc;

    % if the maximum distance is smaller than the tolerance remove vertices
    % between ixs and ixe
    if dmax <= tol;
```

```
        if ixs ~= ixe-1;
            I(ixs+1:ixe-1) = false;
        end
    % if not, call simplifyrec for the segments between ixs and ixc (ixc
    % and ixe)
    else
        p   = simplifyrec(p,tol,ixs,ixc);
        p   = simplifyrec(p,tol,ixc,ixe);

    end

end
end
```

```
function [ind, label] = drawline(p1,p2,image_size)
%DRAWLINE Returns the geometric space (matrix indices) occupied by a line
segment
%in a MxN matrix.  Each line segment is defined by two endpoints.
%
%   IND = DRAWLINE(P1, P2, IMAGE_SIZE) returns the matrix indices
%   of the line segment with endpoints p1 and p2.
%    If both points are out of the image boundary no line is drawn and an
error will appear.
%   If only one of the endpoints is out of the image boundary a line is still
drawn.
%
%       ARGUMENT DESCRIPTION:
%                       P1 - set of endpoints (Nx2). ([row column; ...])
%                        P2 - set of endpoints that connect to p1 (Nx2). ([row
column; ...])
%               IMAGE_SIZE - vector containing image matrix dimensions,
%                            where the first element is the number of rows
%                            and the second element is the number of
%                            columns.
%
%       OUTPUT DESCRIPTION:
%                         IND - matrix indices occupied by the line segments.
%                       LABEL - label tag of each line drawn (from 1 to N).
%
%
%       1
%    1 _|_ _ _ _> COLUMNS
%       |_|_|_|_
%       |_|_|_|_
%       |_|_|_|_
%       V
%      ROWS
%
%   Example
%   -------------
%   BW = zeros(250,250);
%   p1 = [10 10; 23 100; -14 -40];
%   p2 = [50 50; 90 100;  50  50];
%   [ind label] = drawline(p1,p2,[250 250]); % OR ...drawline(p2,p1,...
%   BW(ind) = label;
%   figure, imshow(BW,[])
%
% See also line, ind2sub.

% Credits:
% Daniel Simoes Lopes
% ICIST
% Instituto Superior Tecnico - Universidade Tecnica de Lisboa
% danlopes (at) civil ist utl pt
% http://www.civil.ist.utl.pt/~danlopes
%
% June 2007 original version.

% Input verification.
if max(size(p1) ~= size(p2))
    error('The number of points in p1 and p2 must be the same.')
end
if length(size(image_size)) ~= 2
    error('Image size must be bi-dimensional.')
```

```
end

% Cicle for each pair of endpoints.
ind = [];
label = [];
for line_number = 1:size(p1,1)

    % Point coordinates.
    p1r = p1(line_number,1);   p1c = p1(line_number,2);
    p2r = p2(line_number,1);   p2c = p2(line_number,2);

    % Image dimension.
    M = image_size(1); % Number of rows.
    N = image_size(2); % Number of columns.

    % Boundary verification.
    % A- Both points are out of range.
    if  ((p1r < 1 || M < p1r) || (p1c < 1 || N < p1c)) && ...
        ((p2r < 1 || M < p2r) || (p2c < 1 || N < p2c)),
        error(['Both points in line segment n° ', num2str(line_number),...
               ' are out of range. New coordinates are requested to fit',...
               ' the points in image boundaries.'])
    end

    % Reference versors.
    % .....r..c.....
    eN = [-1  0]';
    eE = [ 0  1]';
    eS = [ 1  0]';
    eW = [ 0 -1]';

    % B- One of the points is out of range.
    if (p1r < 1 || M < p1r) || (p1c < 1 || N < p1c) || ...
       (p2r < 1 || M < p2r) || (p2c < 1 || N < p2c),
        % ....Classify the inner and outer point.
        if     (p1r < 1 || M < p1r) || (p1c < 1 || N < p1c)
            out = [p1r; p1c];  in = [p2r; p2c];
        elseif (p2r < 1 || M < p2r) || (p2c < 1 || N < p2c)
            out = [p2r; p2c];  in = [p1r; p1c];
        end
        % Vector defining line segment.
        v = out - in;
        aux = sort(abs(v)); aspect_ratio = aux(1)/aux(2);
        % Vector orientation.
                     north = v'*eN;
        west  = v'*eW;               east  = v'*eE;
                     south = v'*eS;
        % Increments.
        deltaNS = [];
        if north > 0, deltaNS = -1; end
        if south > 0, deltaNS =  1; end
        if isempty(deltaNS), deltaNS = 0; end
        deltaWE = [];
        if east > 0, deltaWE =  1; end
        if west > 0, deltaWE = -1; end
        if isempty(deltaWE), deltaWE = 0; end
        % Matrix subscripts occupied by the line segment.
        if abs(v(1)) >= abs(v(2))
            alpha(1) = in(1); beta(1) = in(2);
```

```
            iter = 1;
            while (1 <= alpha(iter)) && (alpha(iter) <= M) && ...
                    (1 <=  beta(iter)) && (beta(iter)  <= N),
                alpha(iter+1) = alpha(iter) + deltaNS;                % alpha
grows throughout the column direction.
                beta(iter+1)   = beta(iter)   + aspect_ratio*deltaWE;  % beta
grows throughout the row direction.
                iter = iter + 1;
            end
            alpha = round(alpha(1:end-1)); beta = round(beta(1:end-1));
            ind = cat(2,ind,sub2ind(image_size,alpha,beta));
            label = cat(2,label,line_number*ones(1,max(size(alpha))));
        end
        % ...
        if abs(v(1)) < abs(v(2))
            alpha(1) = in(2); beta(1) = in(1);
            iter = 1;
            while (1 <= alpha(iter)) && (alpha(iter) <= N) &&...
                    (1 <=  beta(iter)) && (beta(iter)  <= M),
                alpha(iter+1) = alpha(iter) + deltaWE;                % alpha
grows throughout the row direction.
                beta(iter+1)   = beta(iter)   + aspect_ratio*deltaNS;  % beta
grows throughout the column direction.
                iter = iter + 1;
            end
            alpha = round(alpha(1:end-1)); beta = round(beta(1:end-1));
            ind = cat(2,ind,sub2ind(image_size,beta,alpha));
            label = cat(2,label,line_number*ones(1,max(size(alpha))));
        end
        clear alpha beta
        continue
    end
    % C- Both points are in range.
    in = [p1r; p1c];  out = [p2r; p2c]; % OR in = p2; out = p1;
    % Vector defining line segment.
    v = out - in;
    aux = sort(abs(v)); aspect_ratio = aux(1)/aux(2);
    % Vector orientation.
                  north = v'*eN;
    west  = v'*eW;              east  = v'*eE;
                  south = v'*eS;
    % Increments.
    deltaNS = [];
    if north > 0, deltaNS = -1; end
    if south > 0, deltaNS =  1; end
    if isempty(deltaNS), deltaNS = 0; end
    deltaWE = [];
    if east > 0, deltaWE =  1; end
    if west > 0, deltaWE = -1; end
    if isempty(deltaWE), deltaWE = 0; end
    % Matrix subscripts occupied by the line segment.
    row_range = sort([p1r p2r]);
    col_range = sort([p1c p2c]);
    if abs(v(1)) >= abs(v(2))
        alpha(1) = in(1); beta(1) = in(2);
        iter = 1;
        while (row_range(1) <= alpha(iter)) && (alpha(iter) <= row_range(2))
&& ...
                (col_range(1)   <=      beta(iter))   &&   (beta(iter)      <=
col_range(2)),
```

```
            alpha(iter+1) = alpha(iter) + deltaNS;               % alpha grows
throughout the column direction.
            beta(iter+1)  = beta(iter)  + aspect_ratio*deltaWE; % beta grows
throughout the row direction.
            iter = iter + 1;
        end
        alpha = round(alpha(1:end-1)); beta = round(beta(1:end-1));
        ind = cat(2,ind,sub2ind(image_size,alpha,beta));
        label = cat(2,label,line_number*ones(1,max(size(alpha))));
    end
    % ...
    if abs(v(1)) < abs(v(2))
        alpha(1) = in(2); beta(1) = in(1);
        iter = 1;
        while (col_range(1) <= alpha(iter)) && (alpha(iter) <= col_range(2))
&&...
                (row_range(1)   <=      beta(iter))   &&   (beta(iter)      <=
row_range(2)),
            alpha(iter+1) = alpha(iter) + deltaWE;               % alpha grows
throughout the row direction.
            beta(iter+1)  = beta(iter)  + aspect_ratio*deltaNS; % beta grows
throughout the column direction.
            iter = iter + 1;
        end
        alpha = round(alpha(1:end-1)); beta = round(beta(1:end-1));
        ind = cat(2,ind,sub2ind(image_size,beta,alpha));
        label = cat(2,label,line_number*ones(1,max(size(alpha))));
    end
    clear alpha beta
    continue
end
```

Appendix C IDL Scripts

IDL Code provided by D. J. Rowenhorst of the United States Navel Research Laboratory

```
PRO Read_grain_stack, stack, calib, stack_lr, calib_lr
  ; directory that contains the images
  directory       =       'F:\OfficeComputerBackupApril2014\PhD\Serial
Sectioning\Stacks\'
  file = 'Grain 7 layers Interior.labels.tif'
  file2 = 'Grain 7 layers Exterior.labels.tif'
  trash = QUERY_TIFF(directory+file, info)


  ; read in the number of images in the tiff.
  nlayers = info.num_images
  dim = info.dimensions
  ; check and see if we need to v-flip the tiff images.
  ;This is typical in IDL, most image programs put 0,0 in the upper
left
  ; IDL puts 0,0 in the bottom left which for TIFF leads to a flip.
  IF info.orientation EQ 1 THEN rot = 7 ELSE rot = 0
  stack = FLTARR(dim[0], dim[1], nlayers+2)
  stack2 = stack
  ; stack the images, put section 0 at the top, last section on layer
0.
  ; also pad out the top and bottom layers with 0s
  FOR i=0, nlayers-1 DO BEGIN
    stack[*,*,nlayers+1  -  (i+2)]  =  ROTATE(READ_TIFF(directory+file,
image_index=i), rot)
    stack2[*,*,nlayers+1  -  (i+2)] = ROTATE(READ_TIFF(directory+file2,
image_index=i), rot)
  ENDFOR

  ; seqentially order the numbers of the grain IDs
  stack = Reorder(stack)
  PRINT, MAX(stack)
  stack2 = Reorder(stack2)

  ;create a structuring elemement for dilation of the exterior grains
  s1 = FLTARR(3,3)+1 ;[ [0,1,0],[1,1,1],[0,1,0]]
  ; dilate out the exterior grain IDs so that they completely overlap
the layer above and below the interior
  ; grains.
  FOR i=0, nlayers-1 DO BEGIN
    stack[*,*,i]  =  DILATE(stack[*,*,i],  s1,  /gray,  /constrained,
background=0)
    FOR j=0, 30 DO BEGIN
      stack2[*,*,i] = DILATE(stack2[*,*,i], s1, /gray, /constrained,
background=0)
    ENDFOR
  ENDFOR
```

```
  ; now erase any dilation that occured within the interior grains
  stack2 *= stack EQ 0
  ; make the exterior grains all lable greater than the interior
grains
  stack2 += (MAX(stack))*(stack2 GT 0)
  ; combine the stacks.
  stack += stack2

  ;combine grains 31 32.  Otherwise a false triple junction is formed.
  wh = WHERE(stack EQ 32)
  stack[wh] = 31

  ; record the voxel size
  calib = [.18, .18, 2]
  ; going to create two versions, full res and a low res (lr).
  calib_lr = [.18*3, .18*3, 2]

  stack_lr = stack[0:*:3, 0:*:3,*]

END
```

```
;function reorder, x, remap = remap
;; quick and dirty function to recast integer data.
;; similar to the ranks function...
;
;srt = sort(x)
;szx = size(x, /dim)
;n_x = n_elements(x)
;recast = ulonarr(szx)
;
;For i=1ll, n_x-1 Do Begin
;   If x[srt[i]] eq x[srt[i-1]] then recast[srt[i]] = recast[srt[i-1]]
else recast[srt[i]] = recast[srt[i-1]]+1
;EndFor
;
;remap = x[uniq(x,srt)]
;return, recast
;
;end

; new version is faster 10/6/2011

FUNCTION Reorder, x, remap = remap
  ; quick and dirty function to recast integer data.
  ; similar to the value_loca function...

  srt = SORT(x)
  srtx = x[srt]
  remap = srtx[Uniq(srtx)]
  IF N_ELEMENTS(remap) GT 1 THEN $
    RETURN, VALUE_LOCATE(remap, x) $
  ELSE $
    RETURN, LONARR(N_ELEMENTS(x))

END
```

```
;+
; NAME:
;       SORT_ND
;
; PURPOSE:
;
;       Efficiently perform an N-dimensional sort along any dimension
;       of an array.
;
; CALLING SEQUENCE:
;
;       inds=sort_nd(array,dimension)
;
; INPUTS:
;
;       array: An array of at least 2 dimensions to sort.
;
;       dimension: The dimension along which to sort, starting at 1
;          (1:rows, 2:columns, ...).
;
; OUTPUTS:
;
;       inds: An index array with the same dimensions as the input
;          array, containing the (1D) sorted indices.  Can be used
;          directly to index the arary (ala SORT).
;
; EXAMPLE:
;
;       a=randomu(sd,5,4,3,2)
;       sorted=a[sort_nd(a,2)]
;
; SEE ALSO:
;
;       HISTOGRAM
;
; MODIFICATION HISTORY:
;
;       Tue Aug 22 15:51:12 2006, J.D. Smith <jdsmith@as.arizona.edu>
;
;          Written, based on discussion on c.l.i-p, 08/2006.
;
;-
;###########################################################################
#########
;
; LICENSE
;
;  Copyright (C) 2006 J.D. Smith
;
;  This file is free software; you can redistribute it and/or modify
;  it under the terms of the GNU General Public License as published
;  by the Free Software Foundation; either version 2, or (at your
;  option) any later version.
;
;  This file is distributed in the hope that it will be useful, but
```

```
;  WITHOUT ANY WARRANTY; without even the implied warranty of
;  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
;  General Public License for more details.
;
;  You should have received a copy of the GNU General Public License
;  along with this file; see the file COPYING.  If not, write to the
;  Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor,
;  Boston, MA 02110-1301, USA.
;
;#####################################################################
#########

function sort_nd, array, dimension
  sz=size(array,/DIMENSIONS)
  ndim=n_elements(sz)
  s=sort(array)

  if dimension eq 1 then begin  ; mark along dimension with index
     inds=s/sz[0]
  endif else begin
     p=product(sz,/CUMULATIVE,/PRESERVE_TYPE)
     inds=s mod p[dimension-2]
     if dimension lt ndim then inds+=s/p[dimension-1]*p[dimension-2]
  endelse

  h=histogram(inds,REVERSE_INDICES=ri)
  ri=s[ri[n_elements(temporary(h))+1:*]]
  if dimension eq 1 then return,reform(ri,sz,/OVERWRITE) $
  else begin                ; target dimension is collected to front,
rearrange it
     t=[dimension-1,where(lindgen(ndim) ne dimension-1)]
     ri=reform(ri,sz[t],/OVERWRITE)
     return,transpose(ri,sort(t))
  endelse
end
```

```
FUNCTION stereo_proj, x, y, z, Inverse=inverse, Negative=negative,
ABSOLUTE=absolute

szx = Size(x)

IF KEYWORD_SET(Inverse) NE 1 THEN BEGIN
;This will be the forward projection.
     IF N_ELEMENTS(y) EQ 0 THEN BEGIN
               yt = REFORM(x[1,*], N_ELEMENTS(x[1,*]) )
               zt = REFORM(x[2,*], N_ELEMENTS(x[2,*]) )
               xt = REFORM(x[0,*], N_ELEMENTS(x[0,*]) )
     ENDIF ELSE BEGIN
               xt = REFORM(x, N_ELEMENTS(x) )
               yt = REFORM(y, N_ELEMENTS(x) )
               zt = REFORM(z, N_ELEMENTS(x) )
     ENDELSE

     IF Keyword_Set(Absolute) THEN BEGIN
               XP = (2 * xt) / (1 + ABS(zt))
               YP = (2 * yt) / (1 + ABS(zt))
     ENDIF ELSE BEGIN
          IF Keyword_Set(negative) THEN BEGIN
               whz = Where(zt LE 1.e-5)
               XP = -(2 * xt[whz]) / (1 - (zt[whz]))
               YP = (2 * yt[whz]) / (1 - (zt[whz]))
          ENDIF ELSE BEGIN
               whz = Where(zt GE -1.e-5)
               XP = (2 * xt[whz]) / (1 + (zt[whz]))
               YP = (2 * yt[whz]) / (1 + (zt[whz]))
          ENDELSE
     ENDELSE

     RETURN, TRANSPOSE([[xp], [yp]])

ENDIF  ELSE BEGIN
;This will be the inverse projection

     IF N_ELEMENTS(y) EQ 0 THEN BEGIN

               yt = REFORM(x[1,*], N_ELEMENTS(x[1,*]) )
               xt = REFORM(x[0,*], N_ELEMENTS(x[0,*]) )

     ENDIF ELSE BEGIN
               xt = REFORM(x, N_ELEMENTS(x) )
               yt = REFORM(y, N_ELEMENTS(x) )
     ENDELSE


     zi = (4. - (xt^2.+ yt^2.))/(4. + (xt^2.+ yt^2.))

     xi = (zi+1.)*(xt/2.)

     yi = (zi+1.)*(yt/2.)
```

```
      IF KEYWORD_SET(Negative) THEN zi = -TEMPORARY(zi)

      RETURN, Transpose([[xi],[yi],[zi]])

ENDELSE

END
```

```
PRO Volume_mesh, volume, v, p , regionID, node_type, vp, voxel_size =
voxszin, no_zero_mesh=no_zero_mesh

  IF N_ELEMENTS(voxszin) GE 3 THEN voxsz = voxszin[0:2] ELSE voxsz =
[1.,1.,1.]

  szvol0 = SIZE(volume)

  vol = LONARR(szvol0[1]+2, szvol0[2]+2,szvol0[3]+2)-1
  IF KEYWORD_SET(no_zero_mesh) THEN $
    vol[1,1,1] = volume-1 ELSE $
    vol[1,1,1] = volume;-MIN(volume)

  szvol = SIZE(vol)
  dim = szvol[1:3]
  dim_p1 = dim+1


  whx = WHERE(vol NE SHIFT(vol, -1, 0, 0), countx)
  IF countx GT 0 THEN BEGIN
    whxyz = Array_indices(dim, whx, /dimensions)
    v0 = whxyz + REBIN([1,0,0], 3, countx)
    v1 = v0 + REBIN([0,1,0], 3, countx)
    v2 = v0 + REBIN([0,0,1], 3, countx)
    v3 = v0 + REBIN([0,1,1], 3, countx)

    regionIDx   =   [TRANSPOSE(Flat(vol[whxyz[0,*],whxyz[1,*],whxyz[2,*]
])), TRANSPOSE(Flat(vol[v0[0,*],v0[1,*],v0[2,*] ]))]
    regionIDx = REFORM(regionIDx, 2, countx, 1)
    regionIDx = REBIN(regionIDx, 2, countx, 2, /sam)
    regionIDx = REFORM(regionIDx, 2, 2*countx)

    v0 = v0[0,*] + v0[1,*]*(dim_p1[0]) + v0[2,*]*(dim_p1[0]*dim_p1[1])
    v1 = v1[0,*] + v1[1,*]*(dim_p1[0]) + v1[2,*]*(dim_p1[0]*dim_p1[1])
    v2 = v2[0,*] + v2[1,*]*(dim_p1[0]) + v2[2,*]*(dim_p1[0]*dim_p1[1])
    v3 = v3[0,*] + v3[1,*]*(dim_p1[0]) + v3[2,*]*(dim_p1[0]*dim_p1[1])

    trix = [[v0, v1, v2],[v2,v1,v3]]

  ENDIF
  whx = 0

  why = WHERE(vol NE SHIFT(vol, 0, -1, 0), county)
  IF county GT 0 THEN BEGIN
    whxyz = Array_indices(dim, why, /dimensions)
    v0 = whxyz + REBIN([0,1,0], 3, county)
    v1 = whxyz + REBIN([0,1,1], 3, county)
    v2 = whxyz  + REBIN([1,1,0], 3, county)
    v3 = whxyz  + REBIN([1,1,1], 3, county)

    regionIDy   =   [TRANSPOSE(Flat(vol[whxyz[0,*],whxyz[1,*],whxyz[2,*]
])), TRANSPOSE(Flat(vol[v0[0,*],v0[1,*],v0[2,*] ]))]
    regionIDy = REFORM(regionIDy, 2, county, 1)
    regionIDy = REBIN(regionIDy, 2, county, 2, /sam)
    regionIDy = REFORM(regionIDy, 2, 2*county)
```

```
    v0 = v0[0,*] + v0[1,*]*(dim_p1[0]) + v0[2,*]*(dim_p1[0]*dim_p1[1])
    v1 = v1[0,*] + v1[1,*]*(dim_p1[0]) + v1[2,*]*(dim_p1[0]*dim_p1[1])
    v2 = v2[0,*] + v2[1,*]*(dim_p1[0]) + v2[2,*]*(dim_p1[0]*dim_p1[1])
    v3 = v3[0,*] + v3[1,*]*(dim_p1[0]) + v3[2,*]*(dim_p1[0]*dim_p1[1])

    triy = [[v0, v1, v2],[v2,v1,v3]]

  ENDIF

  whz = WHERE(vol NE SHIFT(vol, 0, 0, -1), countz)
  IF countz GT 0 THEN BEGIN
    whxyz = Array_indices(dim, whz, /dimensions)
    v0 = whxyz + REBIN([0,0,1], 3, countz)
    v1 = whxyz + REBIN([1,0,1], 3, countz)
    v2 = whxyz  + REBIN([0,1,1], 3, countz)
    v3 = whxyz  + REBIN([1,1,1], 3, countz)

    regionIDz  =  [TRANSPOSE(Flat(vol[whxyz[0,*],whxyz[1,*],whxyz[2,*]
])), TRANSPOSE(Flat(vol[v0[0,*],v0[1,*],v0[2,*] ]))]
    regionIDz = REFORM(regionIDz, 2, countz, 1)
    regionIDz = REBIN(regionIDz, 2, countz, 2, /sam)
    regionIDz = REFORM(regionIDz, 2, 2*countz)

    v0 = v0[0,*] + v0[1,*]*(dim_p1[0]) + v0[2,*]*(dim_p1[0]*dim_p1[1])
    v1 = v1[0,*] + v1[1,*]*(dim_p1[0]) + v1[2,*]*(dim_p1[0]*dim_p1[1])
    v2 = v2[0,*] + v2[1,*]*(dim_p1[0]) + v2[2,*]*(dim_p1[0]*dim_p1[1])
    v3 = v3[0,*] + v3[1,*]*(dim_p1[0]) + v3[2,*]*(dim_p1[0]*dim_p1[1])

    triz = [[v0, v1, v2],[v2,v1,v3]]

  ENDIF

  ; now combine the tree sets of triangles

  tri0 =  [[trix],[triy],[triz], [trix[[0,2,1],*]], [triy[[0,2,1],*]],
[triz[[0,2,1],*]]]

  regionID  =  [[regionIDx],[regionIDy],[regionIDz],[regionIDx[[1,0],*]
], [regionIDy[[1,0],*] ], [regionIDz[[1,0],*] ] ]

  wh = WHERE(regionID[0,*] NE -1, count)
  IF count GT 0 THEN BEGIN
    tri0 = tri0[*, wh]
    regionID = regionID[*,wh]
  ENDIF

  ntri = N_ELEMENTS(tri0)/3l
  uvert = Unique(tri0)
  nvert = N_ELEMENTS(uvert)

  v = Array_indices(dim_p1, uvert, /dim)
  v -= 1 ; bounding box
  v -= 0.5; edge of a voxel
  v *= REBIN(voxsz, 3, nvert)
```

```
  tri = VALUE_LOCATE(uvert, tri0)
  tri0 = 0

  srt = SORT(regionID[0,*])
  regionID = regionID[*, srt]
  tri = tri[*, srt]



  ; now calculate the node type for each vertex point
  node_type = BYTARR(nvert)

  histv = HISTOGRAM(tri, reverse_indices=ri)

  FOR i=0l, nvert-1 DO BEGIN
    whrow = FLOOR(ri[ri[i]:ri[i+1]-1]/3)

    id = Unique(regionID[0,whrow])
    node_type[i] = N_ELEMENTS(id)

  ENDFOR
  ri = 0
  histv = 0

  ; define a vp
  histID = HISTOGRAM(regionID[0,*], reverse_indices = ri, loc=loc)
  wh = WHERE(histID GT 0, count)

  vp = LONARR(3, count)
  vp[0,*] = loc[wh]
  vp[1,*] = ri[ri[wh]]*4
  vp[2,*] = ri[ri[wh+1]-1]*4+3

  IF KEYWORD_SET(no_zero_mesh) THEN BEGIN
    vp[0,*] += 1
    regionID += 1
  ENDIF

  p = LONARR(4, ntri)
  p[0,*] = 3
  p[1:3,*] = tri

END
```

```
function crossp_multi, v1, v2

;v1 = Transpose(Temporary(v1))
;v2 = Transpose(Temporary(v2))
;
;v3 = fltarr(N_elements(v1)/3, 3)
;v3[*,0] = v1[*,1]*v2[*,2] - v1[*,2]*v2[*,1]
;v3[*,1] = v1[*,2]*v2[*,0] - v1[*,0]*v2[*,2]
;v3[*,2] = v1[*,0]*v2[*,1] - v1[*,1]*v2[*,0]
;
;v1 = Transpose(Temporary(v1))
;v2 = Transpose(Temporary(v2))
;Return,  Transpose(v3)


v3 = fltarr(3,N_elements(v1)/3)
v3[0,*] = v1[1,*]*v2[2,*] - v1[2,*]*v2[1,*]
v3[1,*] = v1[2,*]*v2[0,*] - v1[0,*]*v2[2,*]
v3[2,*] = v1[0,*]*v2[1,*] - v1[1,*]*v2[0,*]


Return,  v3



end
```

```
FUNCTION Cubic_color2, poles, gamma=gamma, poles_sym=poles_sym

  If n_elements(gamma) eq 0 then gamma=1.0
  ; poles is a 3xN array of normailzed vectors in cartitian
coordinates.
  n_poles = N_elements(poles[0,*])
  ; first place the colors in the 001 101 111 unit triangle
  poles_abs = Abs(poles)
  poles_sym = poles_abs[Sort_nd(poles_abs,1)] ;sorts each triplet in
increasing order
  poles_sym = (poles_sym[[1,0,2],*]) ;makes z component largest, then
x then y.
  ; done placing them in the triangle
  ;poles_sym = Cubicsym(poles)


  ; the r value will be reversed scaled to the angle from the (001)
axis
  ; reversed scaled  -> r proportional to -angle then normalized to be
0-1.0
  ; allowable range is 0 to !pi/4 in the cubic stereographic triangle.
  mx_ang = Acos(sqrt(1./2.)) ; for cubic unit triangle, max angle from
z axis
  mx_ang2 = !pi/4. ; max angle from x-axis
  r = abs((mx_ang - acos(poles_sym[2,*]))/(mx_ang)) ; scale r to 0 to
1


  ; the green will be reversed scaled with the angle from the x -axis,
blue the inverse of that.
  b = Atan(poles_sym[1,*], poles_sym[0,*]) ; 0 to !pi/4
  g = mx_ang2-b ; want the inverse of blue
  b /= mx_ang2 ; scale to 0 to 1
  g /= mx_ang2 ; scale to 0 to 1

  ;set the ratio of r to gb
  b *= 1.-r
  g *= 1.-r

  rgb = [r,g,b]
  szrgb = size(rgb)

  if szrgb[0] eq 1 then rgb = Reform(rgb, 3, 1)

  ; scale it so that the max of each triplet is 255.0
  rgb /= Rebin(Transpose(Max(rgb, dim=1)), 3,n_poles, /sam)
  rgb *= 255.
  rgb = Byte(Round(rgb))

  ; adust the gamma correction to match TSL more closely
  ramp = Bytscl( Findgen(256)^gamma)

  Return, ramp[rgb]

END
```

```
Function CubicSym, xyzin

;9/10/04
;DJR
;Simple program that will take a set normal vectors and apply the
cubic
;x-stal system symmetry to the vectors and reduce them down to the
unit
;triangle
;input: set of normalized vectors xyzin-> [3,N] array
;output: xyzout equivalent unit triangle vectors xyzout-> [3,N] array
; where N is the number of normals.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

nNormals = N_Elements(xyzin[0,*])

temp = ABS(xyzin)

For i=0L, nNormals-1 Do Begin
     temp[*,i] = temp[Sort(temp[*,i]),i]
EndFor

Return, [temp[1,*], temp[0,*], temp[2,*]]


END
```

```
function      djr_pack_verts,      vin,      pin,      vout,      pout,
aux_data_in=aux_data_in, aux_data_out=aux_data_out, v_index = v_index

  if n_elements(uniq(pin[0:*:4])) gt 3 then return, -1 ;check to see
if the polygon list is triangulated
  npoly = n_elements(pin)/4l
  tri = (reform(pin, 4,npoly))[1:3,*] ;extract the triangles



  vkeep = tri[uniq( tri, sort(tri))] ;get the node values of unique
vertex points

  vout = vin[0:2,vkeep] ; grab all the points


  ; this is for additional data that may be associated with the vertex
points
   if n_elements(aux_data_in) gt 0 then begin
    szAux = size(aux_data_in)
    aux_data_out = aux_data_in
    if  szAux[0]  eq  1  then  aux_data_out  =  reform(aux_data_out,  1,
szAux[1], /over)
    szAux = size(aux_data_out)
    if szAux[2] eq n_elements(vin[0,*]) then begin
      aux_data_out = aux_data_out[*, vkeep]
    endif
  endif

  ; format for output
  v_index = vkeep

  hist = histogram(tri, omin=mn, reverse_in = ri)
  wh = where(hist gt 0, whcount)



  for i=0l, whcount-1 do begin
    ind = ri[ri[wh[i]]:ri[wh[i]+1]-1]
    tri[ind] = i
  endfor

  pout = lonarr(4, npoly)+3
  pout[1:3,*] = tri
  return, npoly

end
```

```
FUNCTION DJRgrPolygon::Init, vertex, VERTEX_DATA=vertex_data, $
           OBJECT_DATA=object_data, $
           vd_column=vd_column,$
           od_column=od_column,$
           EULERS = eulers, $
           SYMMETRY = symmetry, $
           REGIONID = regionid, $
           VERTEX_COLORS = vertex_colors, $
            _EXTRA=e

           IF(self->IDLgrPolygon::Init(vertex,  _EXTRA=e)  NE  1)  THEN
RETURN, 0


           IF N_ELEMENTS(EULERS) EQ 3 THEN $
                self.eulers = EULERS ELSE $
                self.eulers = [-1., -1., -1.]

    IF Size(symmetry, /type) EQ 7 THEN $
      self.symmetry = EULERS ELSE $
      self.symmetry = 'CUBIC'

           IF N_ELEMENTS(REGIONID) EQ 1 THEN $
                self.REGIONID = REGIONID ELSE $
                self.REGIONID = 1L

           self.SELECTED = 1B

           IF N_ELEMENTS(vd_column) EQ 1 THEN $
                self.vd_column = vd_column ELSE $
                self.vd_column = -1
           sz = size(*self.data)
           ; get the size of the vertex array
           ; if the vertex_data has the same trailing size as the
           ; trailing size of the vertices, then keep it
           ; if vertex_data is set to a scaler, then clear the data

           IF N_ELEMENTS(VERTEX_DATA) GE 1 THEN BEGIN
                szVD = size(Vertex_Data)
                IF sz[sz[0]] EQ szVD[szVD[0]] THEN BEGIN
                     If szVD[0] GT 1 THEN $
                          self.vertex_data = PTR_NEW(VERTEX_DATA) $
                     ELSE $
                          self.vertex_data         =         PTR_NEW(
Reform(VERTEX_DATA,1,szVD[szVD[0]]) )
                     ; check and see if the vertex data column is
properly defined.
                     If self.vd_column eq -1 then self.vd_column = 0L
                ENDIF ELSE BEGIN
                     self.vertex_data = PTR_NEW(-1)
                ENDELSE
           ENDIF ELSE BEGIN
                     self.vertex_data = PTR_NEW(-1)
           ENDELSE
```

```
           If N_ELEMENTS(object_data) gt 0 THEN BEGIN
                     self.object_data   =   PTR_NEW(Reform(OBJECT_DATA,
n_elements(OBJECT_DATA)))
           ENDIF ELSE BEGIN
                     self.object_data = PTR_NEW(/allocate)
           ENDELSE

           IF N_ELEMENTS(VERTEX_COLORS) GE 1 THEN BEGIN
             szVC = size(vertex_colors)
      self.vertex_colors = PTR_NEW(vertex_colors)
    ENDIF ELSE BEGIN
      self.vertex_colors = PTR_NEW(-1)
    ENDELSE




           IF N_ELEMENTS(od_column) EQ 1 THEN $
                self.od_column = od_column ELSE $
                self.od_column = 0L

           RETURN,1

END
;_____________________________________________________________________
__________

Pro DJRgrPolygon::Cleanup
     Ptr_free, self.vertex_data,self.object_data
     self->IDLgrPolygon::Cleanup
END

;_____________________________________________________________________
__________
Pro DJRgrPolygon::SetProperty, VERTEX_DATA=vertex_data, $
           OBJECT_DATA=object_data, $
           VERTEX_COLORS = vertex_colors, $
           vd_column=vd_column, $
           od_column=od_column, $
           EULERS = eulers, $
           SYMMETRY = symmetry, $
           REGIONID = regionid, $
           SELECTED = SELECTED, $
            _EXTRA=re


           self->IDLgrPolygon::SetProperty, _EXTRA=re
           sz = size(*self.data)
           ; get the size
           ; if the vertex_data has the same trailing size as the
           ; trailing size of the vertices, then keep it
           ; if vertex_data is set to a scaler, then clear the data
           IF N_ELEMENTS(VERTEX_DATA) GT 1 THEN BEGIN
                szVD = size(Vertex_Data)
                IF sz[sz[0]] EQ szVD[szVD[0]] THEN BEGIN
```

```
                    If szVD[0] GT 1 THEN $
                         *self.vertex_data = VERTEX_DATA $
                    ELSE $
                         *self.vertex_data                              =
Reform(VERTEX_DATA,1,szVD[szVD[0]])
                    ; check and see if the vertex data column is
properly defined.
                    If self.vd_column eq -1 then self.vd_column = 0L
               ENDIF
          ENDIF ELSE BEGIN
               IF N_ELEMENTS(vertex_data) eq 1 then $
               *self.vertex_data = -1
          ENDELSE

          IF N_ELEMENTS(vertex_colors) gt 0 THEN $
              *self.vertex_colors = vertex_colors

          If N_ELEMENTS(object_data) gt 0 THEN $
                    *self.object_data = OBJECT_DATA

          IF N_ELEMENTS(vd_column) EQ 1 THEN $
               self.vd_column = vd_column

          IF N_ELEMENTS(od_column) EQ 1 THEN $
               self.od_column = od_column

          IF N_ELEMENTS(REGIONID) EQ 1 THEN $
               self.REGIONID = REGIONID

          IF N_ELEMENTS(EULERS) EQ 3 THEN $
               self.eulers = EULERS

          IF Size(SYMMETRY, /type) EQ 7 THEN $
     self.symmetry = SYMMETRY

          IF N_ELEMENTS(SELECTED) EQ 1 THEN $
               self.SELECTED = SELECTED

END

;_____________________________________________________________________
__________
Pro DJRgrPolygon::GetProperty, VERTEX_DATA=vertex_data, $
          OBJECT_DATA=object_data, $
          VERTEX_COLORS = vertex_colors, $
          vd_column=vd_column, $
          od_column=od_column, $
          vd_ncolumn=vd_ncolumn, $
          od_ncolumn=od_ncolumn, $
          EULERS = eulers, $
          SYMMETRY = symmetry, $
          REGIONID = regionid, $
          SELECTED = selected, $
          _REF_EXTRA=re
```

```
            self->IDLgrPolygon::GetProperty, _EXTRA=re
            VERTEX_DATA=*self.vertex_data
            If (size(*self.object_data))[0] gt 0 then $
                 OBJECT_DATA=*self.object_data
            vd_column=self.vd_column
            od_column=self.od_column
            sz = size(*self.object_data)
            od_ncolumn = sz[1]
            sz = size(*self.vertex_data)
            vd_ncolumn = sz[1]
            vertex_colors = *self.vertex_colors
            EULERS = self.eulers
            SYMMETRY = self.symmetry
            REGIONID = self.regionid
            SELECTED = self.selected



END
;_____________________________________________________________________
__________
FUNCTION DJRgrPolygon::Get_Vert_data_range, ALL=all
      sz = size(*self.vertex_data)
      IF sz[0] EQ 0 THEN Return, -1
      IF Keyword_set(all) THEN BEGIN
             Return, [min(*self.vertex_data, max=mx), mx]
      ENDIF ELSE BEGIN
            index = self.vd_column
            IF (index LT 0) OR (index GE sz[1]) THEN Return, -2 ELSE $
            Return, [min((*self.vertex_data)[index,*], max=mx), mx]

      ENDELSE
END
;_____________________________________________________________________
__________
FUNCTION DJRgrPolygon::Get_obj_data_range

      sz = size(*self.object_data)
      IF sz[0] EQ 0 THEN Return, -1 ELSE $
            Return, [min(*self.object_data, max=mx), mx]
END
;_____________________________________________________________________
__________
FUNCTION DJRgrPolygon::Get_current_obj_data, index, All=all
      sz = n_elements(*self.object_data)
      IF sz EQ 0 THEN Return, !VALUES.f_nan
      IF Keyword_set(all) THEN BEGIN
             Return, *self.object_data
      ENDIF ELSE BEGIN
            IF N_ELEMENTS(index) eq 0 THEN index = self.od_column

            IF (min(index) LT 0) OR (max(index) GE sz) THEN Return,
!VALUES.f_nan ELSE $
                  Return, (*self.object_data)[index]
      ENDELSE
```

```
END
;_____________________________________________________________________
__________
FUNCTION DJRgrPolygon::Get_current_vert_data, index, All=all

      sz = size(*self.vertex_data)
      IF sz[0] EQ 0 THEN Return, -1
      IF Keyword_set(all) THEN BEGIN
            Return, *self.vertex_data
      ENDIF ELSE BEGIN
           IF N_ELEMENTS(index) eq 0 THEN index = self.od_column
           IF (min(index) LT 0) OR (max(index) GE sz[1]) THEN Return,
-2 ELSE $
                Return, (*self.vertex_data)[index,*]
      ENDELSE
END
;_____________________________________________________________________
__________
PRO DJRgrPolygon::VertData2Color, DATARANGE=datarange, ALL=all
      datarange2 = self-> GET_VERT_DATA_RANGE(ALL=all)
      IF (N_ELEMENTS(datarange) lt 2) THEN datarange = datarange2
      IF (N_ELEMENTS(datarange) lt 2) OR (N_ELEMENTS(datarange2) LT 2)
THEN Return
      IF datarange[0] ge datarange[1] THEN Return
      IF    (self.vd_column    lt    0)    OR    (self.vd_column    ge
(size(*self.vertex_data))[1]) THEN RETURN
      Color = Float((*self.vertex_data)[self.vd_column,*])
      Color = (Color>datarange[0]) < datarange[1]
      Color = Float(Color-datarange[0])/ $
                (datarange[1]-datarange[0])*255

      self->SetProperty, vert_colors=color
END
;_____________________________________________________________________
__________
PRO DJRgrPolygon::OBJData2Color, DATARANGE=datarange

      datarange2 = self->GET_OBJ_DATA_RANGE()
      IF (N_ELEMENTS(datarange) LT 2) THEN datarange = datarange2
      IF (N_ELEMENTS(datarange LT 2)) OR (N_ELEMENTS(datarange2) LT 2)
THEN Return
      IF datarange[0] ge datarange[1] THEN Return
      IF    (self.od_column    lt    0)    OR    (self.od_column    ge
n_elements(*self.object_data)) THEN RETURN
      Color = Float((*self.object_data)[self.od_column])
      Color = (Color>datarange[0]) < datarange[1]
      Color = Float(Color-datarange[0])/ $
                (datarange[1]-datarange[0])*255


      self->SetProperty, vert_colors=REBIN([color],2)
END
```

```
;_____________________________________________________________________
__________
Pro DJRgrPolygon__Define
     Compile_OPT hidden
     struct ={  DJRgrPolygon,$
                     INHERITS IDLgrPolygon, $
                     vertex_data: PTR_NEW(), $
                     vertex_colors: PTR_NEW(), $
                     vd_column: -1L, $
                     object_data: PTR_NEW(),  $
                     od_column: 0L, $
                     eulers: [0.,0.,0.], $
                     symmetry: '', $
                     regionid: 0L, $
                     selected: 0B $
                     }
END
```

```
FUNCTION makeEulerRot, phi, theta, psi, Degrees=degrees


;Simple function to make a rotation matrix from euler angles.
;If keyword, /degree is set, then all the euler angles are in degrees,
;not radians.
;
;Syntax:      rot = makeEulerRot(phi, theta,psi, [/degree])
;

IF N_ELEMENTS(phi) EQ 3 THEN BEGIN
psi = phi[2]
theta = phi[1]
ENDIF

IF Keyword_set(Degrees) EQ 1 Then Begin
     phi = phi/!radeg
     theta = theta/!radeg
     psi = psi/!radeg
EndIF


a11 = COS(psi)*COS(phi[0]) - COS(theta)*Sin(phi[0])*SIN(psi)
a12 = COS(psi)*SIN(phi[0]) + COS(theta)*COS(phi[0])*SIN(psi)
a13 = SIN(psi)*sin(theta)

a21 = -SIN(psi)*COS(phi[0]) - COS(theta)*SIN(phi[0])*COS(psi)
a22 = -SIN(psi)*SIN(phi[0]) + COS(theta)*COS(phi[0])*COS(psi)
a23 = COS(psi)*SIN(theta)

a31 = SIN(theta)*SIN(phi[0])
a32 = -SIN(theta)*COS(phi[0])
a33 = COS(theta)

IF Keyword_set(Degrees) EQ 1 Then Begin
     phi = phi*!radeg
     theta = theta*!radeg
     psi = psi*!radeg
EndIF


Return, [[a11, a12, a13], [a21, a22, a23], [a31, a32, a33]]

END
```

```
PRO makeunittri2, finalimage, ALPHA=ALPHA


;Author: DJR 9/10/04
;Make a cubic unit triangle with some cool color mapping
;the output: finalimage -> an color image of the triangle 400 x 400
pixels.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


res = 300

corners = [[0. , 0.],$
      [2.*1./sqrt(3.)/(1.+1./sqrt(3))                                    ,
2.*1./sqrt(3.)/(1.+1./sqrt(3))],$
      [2.*1./sqrt(2.)/(1.+1./sqrt(2)) , 0]]

x1 = findgen(res)/res*corners[0,1]
y1 = x1
x2 = findgen(res)/res*corners[0,2]
y2 = fltarr(res)
x3 = findgen(res)/res*(corners[0,2] - corners[0,1])+corners[0,1]
y3 = SQRT(8. - (x3+2.)^2. )

x = [transpose(corners[0,*]), x1, x2, x3]
y = [transpose(corners[1,*]), y1, y2, y3]


triangulate, x, y, tri

limit = [min(x)<min(y),min(x)<min(y),max(x)>max(y), max(x)>max(y)]

image = trigrid( x, y, fltarr(N_ELEMENTS(x))+1, tri, [0,0], limit, nx
= 150, ny = 150)

wh = Array_indices(image, where(image))
wh = float(wh)

wh[1,*] = wh[1,*]/max(wh[1,*])*corners[1,1]
wh[0,*] = wh[0,*]/max(wh[0,*])*corners[0,2]

xyz = fltarr(3, N_ELEMENTS(wh[0,*]))
xyz[2,*] = (4 - wh[0,*]^2+wh[1,*]^2.)/(4 + wh[0,*]^2+wh[1,*]^2.)
xyz[0,*] = wh[0,*]/2*(xyz[2,*]+1)
xyz[1,*] = wh[1,*]/2*(xyz[2,*]+1)



;apply last little bit of symetry for cubic system
;xyz = xyz[*,Where(xyz[0,*] LT xyz[2,*])]
xyz = CubicSym(xyz)

npoints = N_Elements(xyz[0,*])
```

```
;Get the Color for the normals
;Cubic_Color, xyz, RGB, XYP = xyp

RGB = cubic_color2(xyz, gam=0.7)

xyp = stereo_proj(xyz)

R = RGB[0,*]
G = RGB[1,*]
B = RGB[2,*]

xP = xyp[0,*]
yP = xyp[1,*]

;Get the vertices and polygons for the flat surface.
Triangulate, xP, yP, Tri

z = Fltarr(N_Elements(xP))

p = LonArr( N_Elements(Tri)+N_Elements(Tri[0,*]))


xsize = Fix(400)
ysize = Fix( xsize* (Max(yp)-Min(yp))/(Max(xp)-Min(xp)) )



image = Bytarr(3,xsize,ysize)

image[0,*,*] = TriGrid(xP,yP,R, tri, NX = xsize, NY=ysize)
image[1,*,*] = TriGrid(xP,yP,G, tri, NX = xsize, NY=ysize)
image[2,*,*] = TriGrid(xP,yP,B, tri, NX = xsize, NY=ysize)


;tv, image, true=1

;if an alpha channel is desired then:
If Keyword_Set(Alpha) Then Begin
finalimage = Bytarr(4,xsize,ysize)
finalimage[0:2,*,*] = image
finalimage[3,*,*] = 255B*( (FIX(image[0,*,*]) + FIX(image[1,*,*]) +
Fix(image[2,*,*])) GT 0)

;For no alpha channel
EndIF Else Begin
finalimage = image
EndElse

END
```

```
Function  Mesh_grain_smooth_shared4,  v,  p,  vertex_type,  ptriline,
out=out
  vsm = v
  nvert = N_elements(v)/3l
  ntri = N_elements(p)/4l
  IF ( (nvert GT 40) AND (Min(vertex_type) NE Max(vertex_type)) ) THEN
BEGIN
    tri = (Reform(p, 4l, ntri))[1:3,*]
    quadpts = Where(vertex_type GT 3, nquadpts)
    faces = Where(vertex_type EQ 2, complement = trijunct)
    ntript = n_elements(trijunct)
    nfacept = n_elements(faces)

    ;make a polygon list that contains only the triple lines
    tri_triline = vertex_type[tri] ge 3
    count = total(tri_triline, 1, /int)
    wh_triline = where(count ge 2, ntri_edge)
    tri_triline = tri[*,wh_triline]
    vtype_triline = vertex_type[tri_triline]
    ;count = count[wh_triline]
    ;ptriline = lonarr(total(count, /int)+n_elements(count))
    ptriline = lonarr(4, ntri_edge)+3
    ptriline[1:3,*] = tri_triline




    ; first smooth the location of the quat points using only the
triple lines
    vsm2 = mesh_smooth(vsm, ptriline, fixed_vert = trijunct, lambda =
0.5, iterations=5)

    ; now smooth the triple lines using only the triple line polygon
list
    ; holding 3% of the triple line points constant on each interation

    ranpts0 = lonarr(ntript)
    stride = 25; CEIL(1.0/0.1)
    ranpts0[0:ntript-1:stride] = 1

    out = fltarr(2, nvert)

    FOR j=1,2 DO BEGIN
      ranshft = stride/j;randomu(seed, 1)*ntript
      ranpts = Where(shift(ranpts0, ranshft))
      ;ranpts = Round(Randomu(seed, nranpts)*(nvert-1))


      fxpts  =  Where(Histogram([trijunct[ranpts],  quadpts],  omin=mn,
min=0))+mn
      ;fxpts = trijunct
      out[0,*] = 0
      out[0,fxpts] = 1

     ;
```

```
     ; vsm2 = Mesh_smooth(vsm2,ptriline, LAMBDA =0.25, fixed_vert =
fxpts, iter=1)
     ;
     vsm2 = Mesh_smooth(vsm2,ptriline, LAMBDA =1, fixed_vert = fxpts,
iter=20)
     ;

    ENDFOR

    wh = where(out[0,*] eq 0)
    fxpts = Where(Histogram([wh, quadpts], omin=mn, min=0))+mn
    vsm2 = Mesh_smooth(vsm2,ptriline, LAMBDA =0.5, fixed_vert = fxpts,
iter=3)
    whsm2 = where(vtype_triline ge 3)
    vsm[*,tri_triline[whsm2]] = vsm2[*,tri_triline[whsm2]]

    ; now smooth the faces
    ranpts0 = lonarr(nfacept)
    stride = 10; how often to put in a fix point
    ranpts0[0:nfacept-1:stride] = 1
    FOR j=1,30 DO BEGIN

      ranshft =   randomu(seed, 1)*(nfacept) ; randomly shift the
starting location of the fix points
      ranpts = Where(shift(ranpts0, ranshft))

      ;always fix the quad pts and tri lines
      fxpts  =  Where(Histogram([faces[ranpts],  quadpts,  trijunct],
omin=mn, min=0))+mn
      out[1,*] = 0
      out[1,fxpts] = 1

     vsm  =  Mesh_smooth(vsm,p,  LAMBDA  =0.07,  fixed_vert  =  fxpts,
iter=30)


    ENDFOR
    ; now smooth the points that were the last to be fixed.
    wh = where(out[1,*] eq 0)
    fxpts  =  Where(Histogram([wh,  quadpts,  trijunct],  omin=mn,
min=0))+mn
    vsm = Mesh_smooth(vsm,p, LAMBDA =0.5, fixed_vert = fxpts, iter=2)

  ENDIF
  return, vsm
END
```

```
Pro NORM_AREA_CALC2, v, p, tri_norm, tri_area, triloc

;Assume everything is a triangle, not a polygon
p = Reform(p,4, N_ELEMENTS(p)/4l, /over)
;temp = Mesh_Decimate(v,p,p2, VERTICES=v2, PERCENT_VERTICES=100)

;v = v2
;p = p2

;v2 = 0
;p2 = 0

numTri = N_Elements(p)/4
;tri_norm = fltarr(3,numTri)
;tri_area = fltarr(numTri)

;point0 = float(v[*,p[1,*]])
;point1 = float(v[*,p[2,*]])
;point2 = float(v[*,p[3,*]])

vector1 = v[0:2,p[2,*]] - v[0:2,p[1,*]]
vector2 = v[0:2,p[3,*]] - v[0:2,p[1,*]]
;vector1 = point1-point0
;vector2 = point2-point0
;normal  = vector1

tri_norm = CrossP_multi(vector1, vector2)
vector1 = 0
vector2 = 0

tri_area = Sqrt(Total(tri_norm^2.,1))/2.0
tri_norm  /=  Rebin(Reform(tri_area,  1,N_Elements(tri_area)  )*2.0,  3,
N_Elements(tri_area), /sample)

;triloc = fltarr(3, numTri)
triloc = v[*,p[1,*]] + v[*,p[2,*]] + v[*,p[3,*]]
triloc /= 3

p = Reform(p, N_ELEMENTS(p), /over)

END
```

```
;;;
```

```
;---------------------------------------------------------------------
-------
;---------------------------------------------------------------------
-------
;---------------------------------------------------------------------
-------


PRO Polyview_help, sEvent
  ; This event handler is for the ABOUT and HELP options

  ; This is the version history text needed for later
  AboutMessage =["Polyviewer Version 2.0 -Authors: Roberto Mendoza and
Dave Rowenhorst"]

  HelpMessage = ["If you have gotten this far you probably don't need
much help. Just press all the buttons.",$
    "OK- a little help.  In the object window, the left mouse button
rotates or translates.",$
    "The middle button selects an object so that you can change its
properties. Middle clicking in the",$
    "background (anything that is not an object, will select all
objects (this is the Default). ",$
    "The left button makes the object disappear. "]

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy

  ; What information is wanted?
  Widget_control, sEvent.id, GET_UValue=want

  CASE want OF

    'ABOUT': BEGIN
      BEEP
      RESULT    =    DIALOG_MESSAGE    (AboutMessage,    /INFORMATION,
TITLE='ABOUT')
    END

    'HELP': BEGIN
      BEEP
      RESULT    =    DIALOG_MESSAGE    (HelpMessage,    /INFORMATION,
TITLE='HELP')
    END


  ENDCASE

  ;Put the info structure back.
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy

END


;---------------------------------------------------------------------
-------
```

```
PRO Polyview_event, sEvent

  Widget_control, sEvent.id, GET_UVALUE=uval


  ; Handle KILL requests.
  IF Tag_names(sEvent, /STRUCTURE_NAME) EQ 'WIDGET_KILL_REQUEST' THEN
BEGIN
    Widget_control, sEvent.top, GET_UVALUE=sState
    ;OBJ_DESTROY, sState.oHolder
    Ptr_free, sState.uColor
    Heap_free, sState.oHolder
    Heap_free, sState.oPalette
    Widget_control, sEvent.top, /DESTROY
    Heap_gc
    Return
  ENDIF

  ; Handle other events.
  wYaw = Widget_info(sEvent.top, Find_By_UName='YAW')
  wPitch = Widget_info(sEvent.top, Find_By_UName='PITCH')
  wRoll = Widget_info(sEvent.top, Find_By_UName='ROLL')
  rSlider = Widget_info(sEvent.top, Find_By_UName='RSLIDER')
  gSlider = Widget_info(sEvent.top, Find_By_UName='GSLIDER')
  bSlider = Widget_info(sEvent.top, Find_By_UName='BSLIDER')
  aSlider = Widget_info(sEvent.top, Find_By_UName='ASLIDER')


  CASE uval OF
    'DRAW': BEGIN
      Widget_control, sEvent.top, GET_UVALUE=sState, /NO_COPY

      ; Handle trackball updates.
      Widget_control,                              Widget_info(sEvent.top,
FIND_BY_UNAME='ROT/TRANS'), GET_VALUE=translate
      bHaveTransform  =  sState.oTrack->Update(  sEvent,  TRANSFORM=qmat,
Translate = translate )
      IF (bHaveTransform NE 0) THEN BEGIN
        sState.oGroup->Getproperty, TRANSFORM=t
        t = t#qmat
        sState.oGroup->Setproperty, TRANSFORM=t
        Getypr, t, ypr
        Widget_control, wyaw, Set_Value=ypr[0]
        Widget_control, wpitch, Set_Value=ypr[1]
        Widget_control, wroll, Set_Value=ypr[2]
        sState.oWindow->Draw, sState.oViewGroup
        Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
        Return

      ENDIF

      IF (sEvent.type EQ 4) THEN BEGIN;  Type  4  indicates  an  intial
draw
        Widget_control, /Hourglass
        sState.oWindow->Draw, sState.oViewGroup
```

```
        Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
        Return
      ENDIF

      ; Button press DOWN.
      IF (sEvent.type EQ 0) THEN BEGIN

        CASE sEvent.Press OF
          4: BEGIN ; Right mouse.
            picked  =  sState.oWindow->Select(sState.oView,  [sEvent.x,
sEvent.y])

            IF Obj_valid(picked[0]) GT 0 THEN BEGIN
              IF    (Obj_class(picked[0])    EQ    'DJRGRPOLYGON')    OR
(Obj_class(picked[0]) EQ 'IDLGRPOLYLINE') THEN BEGIN
                picked[0]->Setproperty, Hide = 1

                wh = Where(picked[0] EQ sState.oMyPolygons)

                IF (wh NE -1) THEN BEGIN
                  objshow=Widget_info(sEvent.top,
Find_By_Uname='OBJSHOWHIDE')
                  Widget_control, objshow, Get_Value=currentShow
                  currentShow[0] = 0
                  currentShow[wh+1] = 0
                  Widget_control,objshow, Set_Value = currentShow
                ENDIF
                sState.oWindow->Draw, sstate.oViewGroup
              ENDIF
            ENDIF

            Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
            Return
          END
          1: BEGIN ;LEFT mouse button.
            sEVENT.PRESS = 1b
            IF sState.dragq GE 0 THEN BEGIN
              sState.oWindow->Setproperty, QUALITY=sState.dragq
              Widget_control, sState.wDraw, /DRAW_MOTION

            ENDIF ELSE BEGIN
              sState.oNoAxis-> SetProperty, Hide=1
              ;For i = 0, N_ELEMENTS(sState.oMyPolygons)-1 Do BEGIN
              ;  sState.oMyPolygons[i]->SetProperty, Hide = 1
              ;ENDFOR
              sState.oWindow->Draw, sstate.oViewGroup
              Widget_control, sState.wDraw, /DRAW_MOTION
            ENDELSE
          END
          2: BEGIN ;Middle Button - select an object
            picked  =  sState.oWindow->Select(sState.oView,  [sEvent.x,
sEvent.y])

            IF Obj_valid(picked[0]) GT 0 THEN BEGIN
              IF (Obj_class(picked[0]) EQ 'DJRGRPOLYGON') THEN BEGIN
```

```
                objselect           =           Widget_info(sEvent.top,
Find_By_Uname='OBJSELECT')
                selectaction         =         Widget_info(sEvent.top,
Find_By_Uname='SELECTTYPE')
                Widget_control, selectaction, Get_Value=selecttype
                Widget_control, objselect, Get_Value=currentselect
                wh = Where(picked[0] EQ sState.oMyPolygons)
                CASE selecttype OF
                  0:BEGIN ; Exclusive selection(one at a time)
                  currentSelect[*] = 0
                  currentSelect[wh+1] = 1
                  picked[0]->Getproperty, Color=color, Alpha_Channel =
alpha, Style = style
                  ;WIDGET_CONTROL, sState.labelWid, SET_VALUE=name[0]
                  Widget_control, rSlider, SET_VALUE=color[0]
                  Widget_control, gSlider, SET_VALUE=color[1]
                  Widget_control, bSlider, SET_VALUE=color[2]
                  Widget_control, aSlider, SET_VALUE=alpha*255
                  Widget_control, sState.fillwire, Set_Value = 2-style

                END
                1:BEGIN
                ;currentSelect[wh+1] = 1
                ;currentSelect[*] = 0
                currentSelect[wh+1] = 1
                picked[0]->Getproperty,  Color=color,  Alpha_Channel  =
alpha, Style = style
                ;WIDGET_CONTROL, sState.labelWid, SET_VALUE=name[0]
                Widget_control, rSlider, SET_VALUE=color[0]
                Widget_control, gSlider, SET_VALUE=color[1]
                Widget_control, bSlider, SET_VALUE=color[2]
                Widget_control, aSlider, SET_VALUE=alpha*255
                Widget_control, sState.fillwire, Set_Value = 2-style
              END
              2:BEGIN
              currentSelect[wh+1] = 0
              wh2 = Where(currentSelect GT 0)
              IF wh2[0] GE 0 THEN BEGIN
                sState.oMyPolygons[wh2[0]]->Getproperty,   Color=color,
Alpha_Channel = alpha, Style = style
                ;WIDGET_CONTROL, sState.labelWid, SET_VALUE=name[0]
                Widget_control, rSlider, SET_VALUE=color[0]
                Widget_control, gSlider, SET_VALUE=color[1]
                Widget_control, bSlider, SET_VALUE=color[2]
                Widget_control, aSlider, SET_VALUE=alpha*255
                Widget_control, sState.fillwire, Set_Value = 2-style
              ENDIF
            END
          ENDCASE
          ;stop
          FOR    i=0,    sState.nPart-1    DO    sState.oMyPolygons[i]-
>Setproperty, Select=currentSelect[i+1]
          IF    Total(currentSelect[1:*])    EQ    sState.npart    THEN
currentSelect[0] = 1 ELSE currentSelect[0] = 0
          Widget_control, objselect, set_value=currentSelect
```

```
        ENDIF ELSE BEGIN

        ENDELSE
      ENDIF
    END
    8: BEGIN ; Scroll wheel
      sState.oGroup -> Rotate, [0,1,0], -5
      sState.oGroup->Getproperty, TRANSFORM=t
      Getypr, t, ypr
      Widget_control, wyaw, Set_Value=ypr[0]
      Widget_control, wpitch, Set_Value=ypr[1]
      Widget_control, wroll, Set_Value=ypr[2]
      sState.oWindow->Draw, sstate.oViewGroup
    END
    16: BEGIN ;scroll wheel back
      sState.oGroup -> Rotate, [0,1,0], 5
      sState.oWindow->Draw, sstate.oViewGroup
      sState.oGroup->Getproperty, TRANSFORM=t
      Getypr, t, ypr
      Widget_control, wyaw, Set_Value=ypr[0]
      Widget_control, wpitch, Set_Value=ypr[1]
      Widget_control, wroll, Set_Value=ypr[2]
      sState.oWindow->Draw, sstate.oViewGroup
    END
    ELSE:
  ENDCASE


ENDIF



; Button release.
IF (sEvent.type EQ 1) THEN BEGIN
  IF (sEVENT.Release EQ 1b) THEN BEGIN ;Left mouse release
    IF sState.dragq GE 0 THEN BEGIN
      sState.oWindow->Setproperty, QUALITY=2
      sState.oWindow->Draw, sstate.oViewGroup
    ENDIF ELSE BEGIN
      sState.oNoAxis->Setproperty, Hide = 0
      ;For i=0, N_ELEMENTS(sState.oMyPolygons)-1 Do Begin
      ;
      ;    sState.oMyPolygons[i]->    SetProperty,    Hide    =    1-
sState.viewlist[i+1]
      ;ENDFOR
      sState.oWindow->Draw, sstate.oViewGroup
    ENDELSE
  ENDIF
  sState.btndown = 0b
  Widget_control, sState.wDraw, DRAW_MOTION=0
ENDIF
Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
END

'Yaw': BEGIN
```

```
  Yawpitchroll, sEvent
END
'EDITPROP': Editprop, sEvent

'QUIT': BEGIN
  Widget_control, sEvent.top, GET_UVALUE=sState
  ;OBJ_DESTROY, sState.oHolder
  Heap_free, sState.oPalette
  Heap_free, sState.oHolder
  Ptr_free, sState.uColor
  Widget_control, sEvent.top, /DESTROY
  Heap_gc

  Return
END
ELSE:

ENDCASE

END
;;;
```

```
  wdatatype = Widget_info(sEvent.top, find_by_uname='DATATYPE')
  Widget_control, wdatatype, Get_value=datatype ;0 is object data, 1
is vertex data
  wcolumn = Widget_info(sEvent.top, find_by_uname='COLUMNSELECT')
  column = Widget_info(wcolumn, /combobox_gettext)
  Widget_control, wcolumn, get_value=text
  column = Where(column EQ text)
  Widget_control, mxSlider, Get_value=mx
  Widget_control, mnSlider, Get_value=mn
  redraw = 0
  ;get a scale of the data range
  IF datatype EQ 0 THEN BEGIN
    mxt = sState.objdatarange[1, column]
    mnt = sState.objdatarange[0, column]
  ENDIF ELSE BEGIN
    mxt = sState.vertdatarange[1, column]
    mnt = sState.vertdatarange[0, column]
  ENDELSE
  scale = Abs(mxt-mnt)*1e-5

  CASE uname OF
    'PROPSLIDERMAX': BEGIN
      ;mn = mn < (mx-scale)
      ;if mx lt mxt then begin
      Widget_control, mxSlider, Set_value=mx
      Widget_control, mnSlider, Set_value=mn
      ;endif else begin
      ;    Widget_Control, mxSlider, Set_value=[mx, mn, mx]
      ;    Widget_Control, mnSlider, Set_value=[mn, mn, mx]
      ;endelse
      ;xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
      ;sState.oCAxis->SetProperty, Range=[mn,mx], xcoord_conv=xs
      ;sState.oCAxis->SetProperty,        Tickvalues=FINDGEN(7)/6.*(mx-
mn)+mn, /Use_text_color
      redraw=1
    END
    'PROPSLIDERMIN':BEGIN
    ;mx  = mx > (mn+scale)
    ;if mn gt mnt then begin
    Widget_control, mxSlider, Set_value=mx
    Widget_control, mnSlider, Set_value=mn
    ;endif else begin
    ; Widget_Control, mxSlider, Set_value=[mx, mn, mx]
    ; Widget_Control, mnSlider, Set_value=[mn, mn, mx]
    ;endelse
    ;xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
    ;sState.oCAxis->SetProperty, Range=[mn,mx], xcoord_conv=xs
    ;sState.oCAxis->SetProperty,   Tickvalues=FINDGEN(7)/6.*(mx-mn)+mn,
/Use_text_color
    redraw=1
  END
  'PROPRESET':BEGIN
  ;wcolumn = widget_info(sEvent.top, find_by_uname='COLUMNSELECT')
  ;column = widget_info(wcolumn, /combobox_gettext)
  ;widget_control, wcolumn, get_value=text
```

```
  ;column = Where(column eq text)
  IF datatype EQ 0 THEN BEGIN
    mx = sState.objdatarange[1, column]
    mn = sState.objdatarange[0, column]
  ENDIF ELSE BEGIN
    mx = sState.vertdatarange[1, column]
    mn = sState.vertdatarange[0, column]
  ENDELSE
  Widget_control, mxSlider, Set_value=[mx, mn, mx]
  Widget_control, mnSlider, Set_value=[mn, mn, mx]
  ;xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
  ;sState.oCAxis->SetProperty, Range=[mn,mx], xcoord_conv=xs
  ;sState.oCAxis->SetProperty,     Tickvalues=FINDGEN(7)/6.*(mx-mn)+mn,
/Use_text_color
  redraw = 1
END
'SLIDERRESET':BEGIN
Widget_control, mxSlider, Set_value=[mx, mn, mx]
Widget_control, mnSlider, Set_value=[mn, mn, mx]
;xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
;sState.oCAxis->SetProperty, Range=[mn,mx], xcoord_conv=xs
;sState.oCAxis->SetProperty,       Tickvalues=FINDGEN(7)/6.*(mx-mn)+mn,
/Use_text_color
;legendview = sState.oViewGroup->getbyname('LEGENDVIEW')
;sState.oWindow->Draw, legendview
redraw=1
END
'PALETTE':BEGIN
newpalette = sEvent.index
sState.oPalette -> Loadct, newpalette
redraw = 1
END
'LEGENDON':BEGIN
Widget_control,   Widget_info(sEvent.top,   find_by_uname='LEGENDON'),
get_value=value
sState.oColorLedgend-> Setproperty, hide=1-value[0]
sState.oOIMLedgend-> Setproperty, hide=1-value[1]
redraw=0
legendview = sState.oViewGroup->Getbyname('LEGENDVIEW')
sState.oWindow->Draw, sState.oViewgroup
Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
Return
END
'DATATYPE':BEGIN
  redraw = 1
;wcolumn = widget_info(sEvent.top, find_by_uname='COLUMNSELECT')
;print, sEvent.value
  Case sEvent.value OF
   0:BEGIN
    ncolumns = -1
    FOR i=0, sState.npart-1 DO BEGIN
      sState.oMyPolygons[i]-> GetProperty, od_ncolumn=temp
      ncolumns = ncolumns > temp
    ENDFOR
    Widget_control, wcolumn, Set_Value=Strtrim(Indgen(ncolumns))
```

```
    FOR i=0, sState.nPart-1 DO BEGIN
      sState.oMyPolygons[i] -> GetProperty, Select=flag1
      IF flag1 EQ 1 THEN BEGIN
        sState.oMyPolygons[i]-> SetProperty, od_column = 0
      ENDIF
    ENDFOR
    mx = sState.objDataRange[1, 0]
    mn = sState.objDataRange[0, 0]
    Widget_control, mxSlider, Set_value=[mx, mn, mx]
    Widget_control, mnSlider, Set_value=[mn, mn, mx]
    redraw=1

    END
    1: BEGIN
    sState.oMyPolygons[0]-> GetProperty, vd_ncolumn=ncolumns
    Widget_control, wcolumn, Set_Value=Strtrim(Indgen(ncolumns))
    FOR i=0, sState.nPart-1 DO BEGIN
      sState.oMyPolygons[i] -> GetProperty, Select=flag1
      IF flag1 EQ 1 THEN BEGIN
        sState.oMyPolygons[i]-> SetProperty, vd_column = 0
      ENDIF
    ENDFOR
    mx = sState.vertDataRange[1, 0]
    mn = sState.vertDataRange[0, 0]
    If mx le mn then mx = 1.1*mn
    Widget_control, mxSlider, Set_value=[mx, mn, mx]
    Widget_control, mnSlider, Set_value=[mn, mn, mx]
    END
    2: BEGIN
      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Select=flag1
        IF flag1 EQ 1 THEN BEGIN
            sState.oMyPolygons[i]-> GetProperty, vertex_colors = temp

            sState.oMyPolygons[i]-> SetProperty, vert_colors = temp
            redraw = 0
          ENDIF
       ENDFOR
    END
    ELSE: print, ""
    ENDCASE


END
'COLUMNSELECT':BEGIN
;wcolumn = widget_info(sEvent.top, find_by_uname='COLUMNSELECT')
;column = widget_info(wcolumn, /combobox_gettext)
;widget_control, wcolumn, get_value=text
;column = where(column eq text)
IF datatype EQ 0 THEN BEGIN ; object data column
  FOR i=0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Select=flag1
    IF flag1 EQ 1 THEN BEGIN
      sState.oMyPolygons[i]-> SetProperty, od_column = column
    ENDIF
```

```
  ENDFOR
  mx = sState.objdatarange[1,column]
  mn = sState.objdatarange[0,column]
  If mx le mn then mx = mn + 1.e-6
ENDIF ELSE BEGIN
  FOR i=0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Select=flag1
    IF flag1 EQ 1 THEN BEGIN
      sState.oMyPolygons[i]-> SetProperty, vd_column = column
    ENDIF
  ENDFOR
  mx = sState.vertdatarange[1,column]
  mn = sState.vertdatarange[0,column]
  If mx le mn then mx = mn + 1.e-6
ENDELSE

Widget_control, mxSlider, Set_value=[mx, mn, mx]
Widget_control, mnSlider, Set_value=[mn, mn, mx]
redraw = 1

END
ELSE:
ENDCASE
;stop
xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
sState.oCAxis->Setproperty, Range=[mn,mx], xcoord_conv=xs
sState.oCAxis->Setproperty,            Tickvalues=Findgen(7)/6.*(mx[0]-
mn[0])+mn[0], /Use_text_color

IF redraw EQ 1 THEN BEGIN

  FOR i=0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Select=flag1
    IF flag1 EQ 1 THEN BEGIN
      IF datatype EQ 0 THEN $
        sState.oMyPolygons[i]-> Objdata2Color, datarange=[mn, mx] $
      ELSE $
        sState.oMyPolygons[i]-> Vertdata2Color, datarange=[mn, mx]
    ENDIF
  ENDFOR


ENDIF
sState.oWindow->Draw, sState.oViewGroup
Widget_control, sEvent.top, SET_UVALUE=sState, /NO_COPY
END

;---------------------------------------------------------------------
-------
PRO Polyview_output, sEvent
  ; This event handler creates image files.

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy

  Widget_control, /Hourglass
```

```
  Wait, 0.5

  sState.oWindow->Getproperty, IMAGE_DATA = snapshot
  imageSize = Size(image)

  ; What kind of file is wanted?
  Widget_control, sEvent.id, GET_UValue=whichFileType
  filter = "~/Desktop/"
  CASE whichFileType OF
    'JPEG': BEGIN
      filter = ['*.jpg','*.JPG']
      filename    =    Dialog_pickfile(/Write,    File='my_image.jpg',
Filter=filter, /overwrite)
      IF filename NE '' THEN Write_jpeg, filename, snapshot, True=1,
Quality=100
    END
    'TIFF': BEGIN
      filter = ['*.tiff','*.tif']
      filename    =    Dialog_pickfile(/Write,    File='my_image.tif',
Filter=filter)
      IF filename NE '' THEN Write_tiff, filename, Reverse(snapshot,3)
    END
    'EPS': BEGIN
      filter = ['*.tif','*.TIF']
      filename    =    Dialog_pickfile(/Write,    File='my_image.eps',
Filter=filter, /overwrite)
      IF filename NE '' THEN BEGIN
        sState.oWindow->Getproperty,         Dimensions=viewDimensions,
Units=viewUnits
        ;clipboard = Obj_new('IDLgrClipboard', Dimensions=[3200,3200],
Unit=viewUnits)
        ;clipboard->Draw,     sstate.oViewGroup,     Filename=filename,
postscript = 1
        ;Obj_destroy, clipboard
        clipboard   =   Obj_new('IDLgrBuffer',   Dimensions=[3200,3200],
Unit=viewUnits)
        clipboard->Draw, sstate.oViewGroup
        clipboard->Getproperty, IMAGE_DATA = snapshot
        Obj_destroy, clipboard
        write_tiff, filename, snapshot, orientation=0


      ENDIF
    END
    'Movie': BEGIN

      filename  =  Dialog_pickfile(/Write,  File='0',  Filter='*.jpg',
/overwrite)
      Widget_control, /Hourglass
      IF filename NE '' THEN BEGIN
        count = 0
        step=2
        FOR i=0, 360, step DO BEGIN
          sState.oWindow->Getproperty, IMAGE_DATA = snapshot
```

```
          Write_jpeg,    filename+Strtrim(String(count,    format='(I-
3.3)'),2)+'.jpg' , snapshot, True=1, Quality=100
          count = count+1
          sState.oGroup->Rotate, [0,1,0], step
          sState.oWindow->Draw, sstate.oViewGroup
        ENDFOR
        FOR i=0, 360, step DO BEGIN
          sState.oWindow->Getproperty, IMAGE_DATA = snapshot
          Write_jpeg,    filename+Strtrim(String(count,    format='(I-
3.3)'),2)+'.jpg' , snapshot, True=1, Quality=100
          count = count+1
          sState.oGroup->Rotate, [1,0,0], step
          sState.oWindow->Draw, sstate.oViewGroup
        ENDFOR
      ENDIF
    END

    'MovTime':BEGIN

    filename = Dialog_pickfile(/Write, File='0', Filter='*.jpg')
    Widget_control, /Hourglass
    IF filename NE '' THEN BEGIN
      count = 0
      step=2
      FOR        i=0,        N_elements(sState.oMyPolygons)-1        DO
sState.oMyPolygons[i]->Setproperty, Hide = 1
      FOR i=0, N_elements(sState.oMyPolygons)-1 DO BEGIN
        sState.oMyPolygons[i]->Setproperty, Hide = 0
        sState.oWindow->Draw, sstate.oViewGroup
        count=count+1
        sState.oWindow->Getproperty, IMAGE_DATA = snapshot
        Write_jpeg,   Strcompress(filename+String(count,   format='(I-
4.4)')+'.jpg', /REMOVE_ALL) , snapshot, True=1, Quality=100
        sState.oMyPolygons[i]->Setproperty, Hide = 1
      ENDFOR
    ENDIF
  END

  'RECORD': BEGIN
    Widget_control, sState.recordbutton, Get_Value=value
    temp =''
    IF value EQ 'Start Recording' THEN BEGIN
      file   =   Dialog_pickfile(/Write,   File='0',   Filter='*.jpg',
Get_Path= temp)
      sState.filename=file
      IF sState.filename NE '' THEN BEGIN
        Widget_control, sState.recordbutton, Set_Value='Recording'



      ENDIF
    ENDIF ELSE BEGIN
      Widget_control, sState.recordbutton, Set_Value='Start Recording'
    ENDELSE
  END
```

```
ENDCASE


;Put the info structure back.
Widget_control, sEvent.top, Set_UValue=sState, /No_Copy

END


;---------------------------------------------------------------------
-------
PRO Polyview_properties, sEvent
  ; This event handler to set the graphic properties.

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy

  ; What property is wanted?
  Widget_control, sEvent.id, Get_UValue=newProperty

  CASE newProperty OF

    ;Reset to the standard colors
    'OG': BEGIN
      sState.oView->Setproperty, Color=[0,0,0]
      sState.xAxis->Setproperty, Color=[255,255,0]
      sState.yAxis->Setproperty, Color=[255,255,0]
      sState.zAxis->Setproperty, Color=[255,255,0]
      sState.dragq = 1
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END

    ; Background color.
    'BBLACK': BEGIN
      sState.oView->Setproperty, Color=[0,0,0]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'BWHITE': BEGIN
      sState.oView->Setproperty, Color=[255,255,255]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'BCHARCOAL': BEGIN
      sState.oView->Setproperty, Color=[80,80,80]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'BGRAY': BEGIN
      sState.oView->Setproperty, Color=[135, 135, 135]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
```

```
    ; Axes colors.
    'ABLACK': BEGIN
      sState.xAxis->Setproperty, Color=[0,0,0]
      sState.yAxis->Setproperty, Color=[0,0,0]
      sState.zAxis->Setproperty, Color=[0,0,0]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'AWHITE': BEGIN
      sState.xAxis->Setproperty, Color=[255,255,255]
      sState.yAxis->Setproperty, Color=[255,255,255]
      sState.zAxis->Setproperty, Color=[255,255,255]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'AGREEN': BEGIN
      sState.xAxis->Setproperty, Color=[0,255,0]
      sState.yAxis->Setproperty, Color=[0,255,0]
      sState.zAxis->Setproperty, Color=[0,255,0]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'AYELLOW': BEGIN
      sState.xAxis->Setproperty, Color=[255,255,0]
      sState.yAxis->Setproperty, Color=[255,255,0]
      sState.zAxis->Setproperty, Color=[255,255,0]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END
    'ANAVY': BEGIN
      sState.xAxis->Setproperty, Color=[0, 0, 115]
      sState.yAxis->Setproperty, Color=[0, 0, 115]
      sState.zAxis->Setproperty, Color=[0, 0, 115]
      ; Redraw the graphic.
      sState.oWindow->Draw, sstate.oViewGroup
    END

    ; Drag Quality
    'VERYLOW': sState.dragq = -1
    'LOW': sState.dragq = 0
    'MEDIUM': sState.dragq = 1
    'HIGH': sState.dragq = 2
    'BFC': BEGIN
      uname = Widget_info(sEvent.id, /UNAME)
      CASE UNAME OF
        'BFCNeg': BEGIN
          FOR    i=0,    sState.npart-1    DO    sState.oMyPolygons[i]-
>Setproperty, reject=1
        END
        'BFCPos':BEGIN
        FOR i=0, sState.npart-1 DO sState.oMyPolygons[i]->Setproperty,
reject=2
      END
      'BFCNone':BEGIN
```

```
      FOR i=0, sState.npart-1 DO sState.oMyPolygons[i]->Setproperty,
reject=0
    END
  ENDCASE
  sState.oWindow->Draw, sstate.oViewGroup
END
'AXIS': BEGIN
  sState.xAxis->Getproperty, HIDE=hidden
  CASE hidden OF
    0: BEGIN
      axisString = 'Show Axis'
      sState.xAxis->Setproperty, HIDE=1
      sState.yAxis->Setproperty, HIDE=1
      sState.zAxis->Setproperty, HIDE=1
    END
    1: BEGIN
      axisString = 'Hide Axis'
      sState.xAxis->Setproperty, HIDE=0
      sState.yAxis->Setproperty, HIDE=0
      sState.zAxis->Setproperty, HIDE=0
    END
  ENDCASE
  Widget_control,  Widget_info(sEvent.top,  find_by_uname='AXISBUT'),
SET_VALUE=axisString
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END

; Lighting Schemes
'ONE': BEGIN
  sState.oLOIM->Setproperty, HIDE=1
  sState.oL1->Setproperty, HIDE=1
  sState.oL2->Setproperty, HIDE=0
  sState.lightstatus=[1,0]
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END
'TWO': BEGIN
  sState.oLOIM->Setproperty, HIDE=1
  sState.oL1->Setproperty, HIDE=0
  sState.oL2->Setproperty, HIDE=1
  sState.lightstatus=[0,1]
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END

'THREE': BEGIN
  sState.oLOIM->Setproperty, HIDE=0
  sState.oL1->Setproperty, HIDE=1
  sState.oL2->Setproperty, HIDE=1
  sState.lightstatus=[0,1]
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END
```

```
'Reset':BEGIN
Widget_control,   Widget_info(sEvent.top,   FIND_BY_UNAME='ROT/TRANS'),
Get_Value=value

IF value EQ  0  THEN BEGIN
  sState.oGroup->Setproperty, Transform = sState.origTransform
  Getypr, sState.origTransform, ypr
  Widget_control,     Widget_info(sEvent.top,     FIND_BY_UNAME='YAW'),
Set_Value=ypr[0]
  Widget_control,    Widget_info(sEvent.top,    FIND_BY_UNAME='PITCH'),
Set_Value=ypr[1]
  Widget_control,     Widget_info(sEvent.top,     FIND_BY_UNAME='ROLL'),
Set_Value=ypr[2]
ENDIF ELSE BEGIN
  sState.oGroup->Getproperty, Transform = t
  t[3,*]=[0,0,0,1]
  t[*,3]=[0,0,0,1]
  sState.oGroup->Setproperty, Transform = t
ENDELSE
; Redraw the graphic.
sState.oWindow->Draw, sstate.oViewGroup
END
ENDCASE



;Put the info structure back.
Widget_control, sEvent.top, Set_UValue=sState, /No_Copy

END
;---------------------------------------------------------------------
----------
PRO Polyview_autotrans, sEvent
  ; This event handler to set the graphic properties.

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  aSlider = Widget_info(sEvent.top, Find_By_UName='ASLIDER')
  ; What property is wanted?
  Widget_control, sEvent.id, Get_UValue=newProperty

  CASE newProperty OF
    'brFront' : BEGIN
      FOR i = 0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          obj = sState.oMyPolygons[i]
          wh = Where(sState.oNoAxis->Get(/all,count=count) EQ obj )

          FOR j=0,count-1 DO BEGIN
            sState.oNoAxis->Move, wh, ((wh-1) > 0)
            wh = (wh-1) > 0
          ENDFOR
        ENDIF
      ENDFOR
```

```
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END
'sndBack' : BEGIN
  FOR i = 0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Selected = flag1
    IF Flag1 EQ 1 THEN BEGIN
      obj = sState.oMyPolygons[i]
      wh = Where(sState.oNoAxis->Get(/all, Count=count) EQ obj )

      FOR j=0,count-1 DO BEGIN
        sState.oNoAxis->Move, wh, ((wh+1) < (count-1))
        wh = (wh+1) < (count-1)

      ENDFOR
    ENDIF
  ENDFOR
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup

END
'brForward': BEGIN
  FOR i = 0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Selected = flag1
    IF Flag1 EQ 1 THEN BEGIN
      obj = sState.oMyPolygons[i]
      wh = Where(sState.oNoAxis->Get(/all) EQ obj )
      sState.oNoAxis->Move, wh, (wh-1) > 0
    ENDIF
  ENDFOR
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END
'sndBackward': BEGIN
  FOR i = 0, sState.nPart-1 DO BEGIN
    sState.oMyPolygons[i] -> GetProperty, Selected = flag1
    IF Flag1 EQ 1 THEN BEGIN
      obj = sState.oMyPolygons[i]
      wh = Where(sState.oNoAxis->Get(/all, Count=count) EQ obj )
      sState.oNoAxis->Move, wh, (wh+1) < (count-1)

    ENDIF
  ENDFOR
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup
END


'autotrans':BEGIN

nobj = sstate.npart
pos = -1l
objs = Objarr(1)

micro = sState.oNoAxis->Get(/all)
```

```
    FOR i=0, nobj-1 DO BEGIN
      wh1 = Where(micro EQ sState.omypolygons[i] )
      micro[wh1]-> getproperty, hide=hide, select=select
      IF (select) EQ 1 THEN BEGIN
        pos = [[pos], wh1]
        objs = [[objs], sState.omypolygons[i]]
      ENDIF

    ENDFOR

    IF N_elements(pos) GT 1 THEN BEGIN
      pos = pos[1:*]
      objs = objs[1:*]
      trans = Float(Sort(pos))
      trans = 0.75*trans/Max(trans)+0.25
      trans = Reverse(trans)
      FOR    i=0,    N_elements(trans)-1    DO    objs[i]->Setproperty,
alpha=trans[i]
      obj = objs[0]
      obj->Getproperty, alpha = alpha
      Widget_control, aslider, set_value=255*alpha

    ENDIF

    ; Redraw the graphic.
    sState.oWindow->Draw, sstate.oViewGroup

  END

  ELSE:

ENDCASE
;Put the info structure back.
Widget_control, sEvent.top, Set_UValue=sState, /No_Copy
END
;---------------------------------------------------------------------
-------
PRO Polyview_zoom, sEvent
  ; This is to set the value of zoom
  ;Widget_Control, sEvent.id, Get_UValue=newProperty
  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  Widget_control,  Widget_info(sEvent.top,  Find_By_UNAME='ZOOMSLIDE'),
Get_Value=newProperty

  ; Get the aspect ratio

  aspect = sState.aspect

  ; Get the view and put it in
  Getview, newProperty, aspect, myView
  sState.oView->Setproperty, VIEWPLANE_RECT=myView
  sState.oWindow->Draw, sstate.oViewGroup

  ; Put the info structure back.
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy
```

```
END

;---------------------------------------------------------------------
-------
PRO Getview, zoomFactor, aspect, myView
  ; This function is used to calculate the VIEW used to zoom in and
out

  ; Set up the equation for the zoom
  zeroVal = 2.5;FLOAT(1.9) ; Larger value makes it zoom less
  topVal  = Float(0.1)
  slope   = Float( (topVal-zeroVal)/100.0 )

  sqrt2 = slope * zoomFactor + zeroVal

  myview = [ -sqrt2*0.5, -sqrt2*0.5, sqrt2, sqrt2 ]
  IF (aspect GT 1) THEN BEGIN
    myview[0] = myview[0] - ((aspect-1.0)*myview[2])/2.0
    myview[2] = myview[2] * aspect
  ENDIF ELSE BEGIN
    myview[1] = myview[1] - (((1.0/aspect)-1.0)*myview[3])/2.0
    myview[3] = myview[3] / aspect
  ENDELSE

END


FUNCTION Dave_caps, sEvent

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  Widget_control, sState.wcaps, Get_Value=value
  planes = -1
  IF value[0] EQ 0 THEN planes = sState.clipplanes[*,0]
  IF value[1] EQ 0 THEN planes = sState.clipplanes[*,1]
  IF  (value[0]  EQ  0)  AND  (value[1]  EQ  0)    THEN  planes  =
sState.clipplanes

  FOR  i=0,  sState.npart-1  DO  sState.oMyPolygons[i]->Setproperty,
Clip_Planes=planes
  sState.oWindow->Draw, sstate.oViewGroup


  ; Put the info structure back.
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy
  Return, 1
END

FUNCTION Dave_objpicker, sEvent
  ; This is the control for the Hide/Show Control Box

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  objshow=Widget_info(sEvent.top, Find_By_Uname='OBJSHOWHIDE')
  objselect = Widget_info(sEvent.top, Find_By_Uname='OBJSELECT')
  seclectaction = Widget_info(sEvent.top, Find_By_Uname='SELECTTYPE')
```

```
  Widget_control, sEvent.id, GET_UVALUE=uval
  CASE uval OF
    'objhide':BEGIN
    Widget_control, objshow, Get_Value=currentShow
    button = sEvent.value
    nObj = N_elements(sState.oMyPolygons)
    CASE button OF
      0: BEGIN
        IF currentShow[0] EQ 0 THEN BEGIN
          FOR      i=0,       N_elements(sState.oMyPolygons)-1       DO
sState.oMyPolygons[i]->Setproperty, Hide = 1
          Widget_control, objshow, Set_Value=Intarr(nObj+1)
        ENDIF ELSE BEGIN
          FOR      i=0,       N_elements(sState.oMyPolygons)-1       DO
sState.oMyPolygons[i]->Setproperty, Hide = 0
          ;If total(currentShow[1:*]) eq 0 then begin
          ;      FOR     i=0,     N_ELEMENTS(sState.oMyPolygons)-1     DO
sState.oMyPolygons[i]->SetProperty, Select = 1
          ;Endif
          Widget_control, objshow, Set_Value=Intarr(nObj+1)+1
        ENDELSE
      END
      ELSE:BEGIN
      FOR        i=0,        N_elements(sState.oMyPolygons)-1        DO
sState.oMyPolygons[i]->Setproperty, Hide = 1-currentShow[i+1]
      tot = Total(currentShow[1:*])
      IF tot EQ nObj THEN currentShow[0] = 1 ELSE currentShow[0] = 0
      Widget_control, objshow, Set_Value=currentShow
    END
  ENDCASE

  sState.oWindow->Draw, sstate.oViewGroup
END
'objselect':BEGIN
;Widget_Control, selectaction, Get_Value=selecttype
Widget_control, objselect, Get_Value=currentselect
button = sEvent.value
IF button EQ 0 THEN BEGIN
  IF currentselect[0] EQ 0 THEN BEGIN
    currentselect[*] = 0
  ENDIF ELSE BEGIN
    currentselect[*] = 1
  ENDELSE
ENDIF ELSE BEGIN

ENDELSE

FOR   i=0,   N_elements(currentSelect)-2   DO   sState.oMyPolygons[i]-
>Setproperty, Select = currentSelect[i+1]
IF Total(currentSelect[1:*]) EQ sState.npart THEN currentSelect[0] = 1
ELSE currentSelect[0] = 0
Widget_control, objselect, Set_Value=currentSelect
END
'SELECTACTION':BEGIN
```

```
END
ELSE:BEGIN
END
ENDCASE



; Put the info structure back.
Widget_control, sEvent.top, Set_UValue=sState, /No_Copy
Return, 1
END

;_____________________________________________________________________________
_____________________________________________

PRO Dave_rescale, sEvent
  ; This is to set the value of zoom
  Widget_control, /HOURGLASS
  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  oPart = sState.oMyPolygons[0:sState.nPart-1]
  ;Now Figure out the data ranges
  xrange = [1e9,-1.0]
  yrange = [1e9,-1.0]
  zrange = [1e9,-1.0]

  FOR i=0, N_elements(oPart)-1 DO BEGIN
    oPart[i]->Getproperty, Hide=hide
    IF hide EQ 0 THEN BEGIN
      oPart[i]->Getproperty,   XRANGE=xrangetemp,   YRANGE=yrangetemp,
ZRANGE=zrangetemp
      xrange[0] = xrange[0] < xrangetemp[0]
      yrange[0] = yrange[0] < yrangetemp[0]
      zrange[0] = zrange[0] < zrangetemp[0]

      xrange[1] = xrange[1] > xrangetemp[1]
      yrange[1] = yrange[1] > yrangetemp[1]
      zrange[1] = zrange[1] > zrangetemp[1]
    ENDIF
  ENDFOR

  xmin = xrange[0]
  xmax = xrange[1]
  ymin = yrange[0]
  ymax = yrange[1]
  zmin = zrange[0]
  zmax = zrange[1]

  ; Compute data bounds.
  xmm = xMax - xMin
  ymm = yMax - yMin
  zmm = zMax - zMin
  xyzmm = [xmm, ymm, zmm ]
```

```
  ; Compute coordinate conversion to normalize.
  xyzSpan = Max( xyzmm )
  xs = [0.0,1.0/xyzSpan]
  ys = [0.0,1.0/xyzSpan]
  zs = [0.0,1.0/xyzSpan]

  xs[1] = Min( [xs[1], ys[1], zs[1]] ) *0.9
  ys[1] = xs[1]
  zs[1] = ys[1]

  xs[0] = -0.5*(xMax+xMin)*xs[1]
  ys[0] = -0.5*(yMax+yMin)*ys[1]
  zs[0] = -0.5*(zMax+zMin)*zs[1]

  sState.xaxis->Getproperty, Parent = oParent, TickText= oTickText,
Color=aColor

  sState.xaxis->Setproperty,   Range=xrange-xmin,   Ticklen=0.02/xs[1],
Location=[xmin,ymin,zmin], MAJOR=-1, MINOR=-1
  sState.yaxis->Setproperty,   Range=yrange-ymin,   Ticklen=0.02/xs[1],
Location=[xmin,ymin,zmin], MAJOR=-1, MINOR=-1
  sState.zaxis->Setproperty,   Range=zrange-zmin,   Ticklen=0.02/xs[1],
Location=[xmin,ymax,zmin], MAJOR=-1, MINOR=-1

  sState.xaxis->Getproperty, XCoord_Conv = xcoord
  sState.yaxis->Getproperty, YCoord_Conv = ycoord
  sState.zaxis->Getproperty, ZCoord_Conv = zcoord

  xcoord[0] = xmin;+xcoord[0]
  ycoord[0] = ymin;+ycoord[0]
  zcoord[0] = zmin;+zcoord[0]

  sState.xaxis->Setproperty, XCoord_Conv = xcoord
  sState.yaxis->Setproperty, YCoord_Conv = ycoord
  sState.zaxis->Setproperty, ZCoord_Conv = zcoord

  sState.oMicro->Reset
  sState.oMicro->Scale,     xs[1],ys[1],zs[1]
  sState.oMicro->Translate, xs[0],ys[0],zs[0]

  sState.oWindow->Draw, sState.oViewGroup
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy


END

PRO Getypr, t, ypr

  temp = (t[0:2, 0:2])

  y = Atan(temp[1,0]/temp[0,0]) + !PI*(temp[0,0] LT 0)
  p   =   Atan((-temp[2,0])/Sqrt(temp[2,1]^2.   +   temp[2,2]^2.))   +
!PI*(Sqrt(temp[2,1]^2. + temp[2,2]^2.) LT 0)
  r = Atan(temp[2,1]/temp[2,2]) + !PI*(temp[2,2] LT 0)
```

```
  y = 2*!PI*(y LT 0) + y
  p = 2*!PI*(p LT 0) + p
  r = 2*!PI*(r LT 0) + r
  ypr = [y,p,r]*!radeg
;If Total(finite(ypr)) NE 3 Then Stop

END
PRO Setypr, t, ypr0

  ypr = ypr0/!radeg

  temp    =    [[Cos(ypr[1])*Cos(ypr[0]),Cos(ypr[1])*Sin(ypr[0]),    -
Sin(ypr[1])], $
    [Sin(ypr[2])*Sin(ypr[1])*Cos(ypr[0])-Cos(ypr[2])*Sin(ypr[0]), $
    Sin(ypr[2])*Sin(ypr[1])*Sin(ypr[0])+Cos(ypr[2])*Cos(ypr[0]), $
    Cos(ypr[1])*Sin(ypr[2])],$
    [Cos(ypr[2])*Sin(ypr[1])*Cos(ypr[0])+Sin(ypr[2])*Sin(ypr[0]), $
    Cos(ypr[2])*Sin(ypr[1])*Sin(ypr[0])-Sin(ypr[2])*Cos(ypr[0]), $
    Cos(ypr[1])*Cos(ypr[2])]]

  t[0:2, 0:2] = temp
END
;_____________________________________________________________________
_____________________________________


PRO Dave_2doim_event, sEvent

  Widget_control, sEvent.id, Get_UValue=action
  Widget_control, sEvent.top, Get_UValue=tempPointer

  CASE action OF
    'DONE': BEGIN
      Widget_control, (*tempPointer).table, Get_Value=newVect
      (*tempPointer).newvect = newvect
      Widget_control, sEvent.top, /DESTROY

    END

    'CANCEL':BEGIN

    Widget_control, sEvent.top, /DESTROY
  END
  'TChange':BEGIN

END

ELSE:

ENDCASE

END
```

```
PRO Davecolor_polygons, sEvent


  ; This event handler to set the graphic properties.

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy
  rSlider = Widget_info(sEvent.top, Find_By_UName='RSLIDER')
  gSlider = Widget_info(sEvent.top, Find_By_UName='GSLIDER')
  bSlider = Widget_info(sEvent.top, Find_By_UName='BSLIDER')
  aSlider = Widget_info(sEvent.top, Find_By_UName='ASLIDER')

  ; What new color scheme is wanted?
  Widget_control, sEvent.id, Get_UValue=newProperty
  sState.oOIMLedgend->Setproperty, Hide=1
  sState.oColorLedgend->Setproperty, Hide=1
  Widget_control, rSlider, Sensitive=1
  Widget_control, gSlider, Sensitive=1
  Widget_control, bSlider, Sensitive=1


  sState.oL1->Setproperty, Hide=sState.lightstatus[0]
  sState.oL2->Setproperty, Hide=sState.lightstatus[1]
  sState.oLOIM->Setproperty, Hide = 1

  ;For i=0, sState.nPart-1 Do Begin
  ;        sState.oMyPolygons[i]   ->   SetProperty,   Vert_Colors   =
0,shading=1
  ;EndFor

  CASE newProperty OF

    'defaultC': BEGIN
      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          sState.oMyPolygons[i] -> SetProperty, Color = sState.dColor,
shading=1, vert_colors=-1
          Widget_control, rSlider, SET_VALUE=sState.dColor[0]
          Widget_control, gSlider, SET_VALUE=sState.dColor[1]
          Widget_control, bSlider, SET_VALUE=sState.dColor[2]
        ENDIF
      ENDFOR
    END
    'randomC': BEGIN
      FOR i=0, sState.nPart-1 DO BEGIN
        rColor = Byte(255*Randomu(seed, 3))
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          sState.oMyPolygons[i]   ->   SetProperty,   Color   =   rColor,
shading=1, vert_colors = -1
          Widget_control, rSlider, SET_VALUE=rColor[0]
          Widget_control, gSlider, SET_VALUE=rColor[1]
          Widget_control, bSlider, SET_VALUE=rColor[2]
        ENDIF
      ENDFOR
```

```
    END
    'userC': BEGIN
      FOR i=0, sState.nPart-1 DO BEGIN

        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          sState.oMyPolygons[i]     ->     SetProperty,     Color     =
(*sState.uColor)[1:3,i], shading=1, $
            Alpha    =    (Float(    (*sState.uColor)[4,i])/255.>0.)<1.0,
vert_colors=-1
          Widget_control, rSlider, SET_VALUE=(*sState.uColor)[1,i]
          Widget_control, gSlider, SET_VALUE=(*sState.uColor)[2,i]
          Widget_control, bSlider, SET_VALUE=(*sState.uColor)[3,i]
          Widget_control, aSlider, SET_VALUE=(*sState.uColor)[4,i]
        ENDIF
      ENDFOR
    END

    '2DOIM': BEGIN
      Widget_control, /Hourglass
      ;WIDGET_CONTROL, rSlider, Sensitive=0
      ;WIDGET_CONTROL, gSlider, Sensitive=0
      ;WIDGET_CONTROL, bSlider, Sensitive=0
      ;sState.oL1-> SetProperty, Hide=1
      ;sState.oL2-> SetProperty, Hide=1
      ;sState.oLOIM-> SetProperty, Hide=0

      wOIM2Din     =     Widget_base(Group_Leader=sEvent.top,Column=1,
Event_PRO='Dave_2DOIM_Event', /Floating)
      temp = Widget_label(wOIM2Din, Value='Please enter a direction.')
      temp  =  Widget_label(wOIM2Din,  Value='Color  represents  crystal
direction parallel with direction.')
      inputTable = Widget_table(wOIM2Din, Alignment=1, /Edit, $
        Column_Labels=['x','y','z'],Row_Label=['Direction'],
Value=sState.OIM2DVECT,                                  UVALUE='TChange',
Event_PRO='Dave_2DOIM_Event')

      buttonBase = Widget_base(wOIM2Din, Column=2)
      Done  =  Widget_button(buttonBase,  Value='DONE',  UVALUE='DONE',
EVENT_Pro='DAVE_2DOIM_Event')
      Cancel      =      Widget_button(buttonBase,      Value='Cancel',
UVALUE='CANCEL', EVENT_Pro='DAVE_2DOIM_Event')
      tempstate= {newvect:sState.OIM2DVect, table:inputTable}
      tempPointer = Ptr_new(tempstate)

      Widget_control, wOIM2Din, /REALIZE
      Widget_control, wOIM2Din, Set_UVAL = tempPointer
      Xmanager,        '2D        OIM        Manager',        wOIM2Din,
Event_Handler='Dave_2DOIM_Event'

      tempState.newvect=(*tempPointer).newvect

      Ptr_free, tempPointer
```

```
      tempState.newvect                                                 =
tempState.newVect/Sqrt(Total(tempState.newvect^2.))

      sState.OIM2DVECT = tempState.newvect

      ;newtext = 'Color represents crystal direction parallel with ['
+ $
      ;     String(sState.OIM2DVECT, format = '(2(f5.2, ", "),f5.2 )'
) + ']'
      ;sState.oOrientationText->SetProperty, Strings = newtext


      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i]     ->     GetProperty,     Select=flag1,
Eulers=Euler
        IF flag1 EQ 1 THEN BEGIN
          ;pname = Fix(pname)
          ;wh = where(sState.euler[0,*] eq pname)
          IF (Round(Total(euler)) NE -30) THEN BEGIN
            rotmat = Makeeulerrot(euler)
            normals = Transpose(rotmat##sState.OIM2Dvect)
            normals = Cubicsym(normals)
            ;Cubic_color2, normals, RGB
            rgb = Cubic_color2( normals, gamma = 0.7)
            sState.oMyPolygons[i]   ->   SetProperty,   Color   =   rgb,
vert_colors=-1
          ENDIF ELSE BEGIN
          ;sState.oMyPolygons[i] -> SetProperty, hide = 1
          ;sState.viewList[i+1] = 0
          ;sState.viewList[0] = 0
          ENDELSE
        ENDIF
      ;Widget_Control, sState.objpicker, Set_Value=sState.viewlist
      ENDFOR
    ;sState.oOIMLedgend->SetProperty, Hide=0
    END

    'normalC': BEGIN
      Widget_control, /Hourglass
      ;WIDGET_CONTROL, rSlider, Sensitive=0
      ;WIDGET_CONTROL, gSlider, Sensitive=0
      ;WIDGET_CONTROL, bSlider, Sensitive=0
      ;sState.oL1-> SetProperty, Hide=1
      ;sState.oL2-> SetProperty, Hide=1
      ;sState.oLOIM-> SetProperty, Hide=0

      ;newtext = 'Color represents crystal direction parallel with
each surface polygon normal'
      ;sState.oOrientationText->SetProperty, Strings = newtext

      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i]     ->     GetProperty,     Select=flag1,
Euler=euler
        IF flag1 EQ 1 THEN BEGIN
```

```
          IF (Round(Total(euler)) NE -30) THEN BEGIN
            rotmat = Makeeulerrot(euler)

            sState.oMyPolygons[i] -> GetProperty, data = v, poly=p
            normals = Compute_Mesh_Normals(v, p)
            ;For j = 0l, N_Elements(normals[0,*])-1 Do BEGIN
            ;   normals[*,j] = TRANSPOSE(rotmat##normals[*,j])
            ;ENDFOR
            normals = Transpose(rotmat##Transpose(normals))
            normals = Cubicsym(normals)
            ;Cubic_color2, normals, RGB
            rgb = Cubic_color2(normals, gamma=0.7)
            sState.oMyPolygons[i] -> SetProperty, Vert_colors = RGB,
shading=1
          ENDIF ELSE BEGIN
          ;sState.oMyPolygons[i] -> SetProperty, vert_colors=-1

          ENDELSE
        ENDIF

      ENDFOR
    ;sState.oOIMLedgend->SetProperty, Hide=0
    END

    'normalC2': BEGIN
      Widget_control, /Hourglass
      ;WIDGET_CONTROL, rSlider, Sensitive=0
      ;WIDGET_CONTROL, gSlider, Sensitive=0
      ;WIDGET_CONTROL, bSlider, Sensitive=0

      ;newtext = 'Color represents interface direction parallel with
each surface polygon normal'
      ;sState.oOrientationText->SetProperty, Strings = newtext

      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          sState.oMyPolygons[i] -> GetProperty, normals=normals
          ;normals = Compute_Mesh_Normals(v, p)
          normals *= 0.5
          normals = Stereo_proj(Temporary(normals), /ABSOLUTE)
          normals = Cv_coord(from_rect=normals, /to_polar, /degrees)
          hsv = Fltarr(3, N_elements(normals)/2)+0.95
          hsv[0:1,*] =  normals

          ;rgb = bytarr(3, n_elements(hsv)/3)
          Color_convert,  hsv[0,*],  hsv[1,*],  hsv[2,*],  r,  g,  b,
/hsv_rgb
          rgb = [r,g,b]

          sState.oMyPolygons[i] -> SetProperty, Vert_colors = rgb,
shading=1
        ENDIF
      ENDFOR
```

```
    END

    'wire': BEGIN
      FOR i=0, sState.nPart-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN BEGIN
          sState.oMyPolygons[i] -> GetProperty, Style=style
          IF style GE 2 THEN style = -1
          sState.oMyPolygons[i] -> SetProperty, Style = (style+1)
        ENDIF
      ENDFOR

    END

    'colorSlider': BEGIN
      Widget_control, rSlider, GET_VALUE=newRED
      Widget_control, gSlider, GET_VALUE=newGreen
      Widget_control, bSlider, GET_VALUE=newBlue

      newColor = [newRed, newGreen, newBlue]
      FOR i=0, (sState.nPart)-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN $
          sState.oMyPolygons[i]  ->  SetProperty,  Color  =  newColor,
vert_color=0
      ENDFOR

    END
    'alphaSlider': BEGIN
      Widget_control, aSlider, Get_Value=newAlpha
      FOR i=0, (sState.nPart)-1 DO BEGIN
        sState.oMyPolygons[i] -> GetProperty, Selected = flag1
        IF Flag1 EQ 1 THEN $
          sState.oMyPolygons[i] -> SetProperty, Alpha=newAlpha/255.
      ENDFOR

    END
    'OBJECTPROPERTY':BEGIN

    FOR i=0, sState.nPart-1 DO BEGIN
      sState.oMyPolygons[i] -> GetProperty, Select=flag1
      IF flag1 EQ 1 THEN BEGIN
        sState.oMyPolygons[i]->                          Objdata2Color,
datarange=sState.objDataRange
      ENDIF
    ENDFOR

  END

ENDCASE

; Redraw the graphic.
sState.oWindow->Draw, sstate.oViewGroup

; Put the info structure back.
```

```
Widget_control, sEvent.top, Set_UValue=sState, /No_Copy


END


FUNCTION Dave_translate, sEvent

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy

  Widget_control,  Widget_info(sEvent.top,  FIND_BY_UNAME='ROT/TRANS'),
GET_VALUE=translate

  ;sState.translate=translate

  ; Put the info structure back.
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy

END


FUNCTION Dave_fillwire, sEvent

  Widget_control, sEvent.top, Get_UValue=sState, /No_Copy

  Widget_control, sState.FillWire, GET_VALUE=style
  style = 2-style

  ;wh = Where(sState.CurrentOBJ NE OBJ_NEW(), count)
  FOR i=0, sState.npart-1 DO BEGIN
    sState.oMyPolygons[i]-> GetProperty, Select=flag1
    IF flag1 EQ 1 THEN $
      sState.oMyPolygons[i] -> SetProperty, STYLE=style
  ENDFOR
  ; Redraw the graphic.
  sState.oWindow->Draw, sstate.oViewGroup

  ; Put the info structure back.
  Widget_control, sEvent.top, Set_UValue=sState, /No_Copy

END


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
PRO Polyviewer, superV, superP, vpLookup, PLIST = plist2, euler=euler,
Color=uColor, Planes=planes, Points=points, $
    Vertex_colors=vertex_colors,           OBJECT_LABEL=object_label2,
XDIM=xdim, YDIM=ydim, $
    OBJECT_DATA = object_data, Vertex_data=vertex_data, $
    oMicro = oMicro, $
    initialview =initialview , noaxis=noaxis, iansfile = iansfile

  ;DJR
  ;12/02/09
  ; This program was built to examine many surface objects at once.
It has gotten a bit complicated.




  szV = Size(superV)
  szP = Size(superP)
  szVD = Size(vertex_data)

  ; check some sizes of the input arrays.
  IF    N_elements(vpLookup)    EQ    0    THEN    vpLookup=[1L,    0,
N_elements(superP)-1, 0, N_elements(superV)/3-1]

  ;Set up the display window size relative to the screen size.
  IF (N_elements(xdim) EQ 1) AND (N_elements(ydim) EQ 1) THEN BEGIN
    IF (xdim LT 100) OR (ydim LT 100) THEN BEGIN
      Device, GET_SCREEN_SIZE=scr
      ydim = scr[1] * 0.8; * 0.85
      xdim = ydim   * 1; * 0.85
    ENDIF
  ENDIF ELSE BEGIN
    Device, GET_SCREEN_SIZE=scr
    ydim = scr[1] * 0.8; * 0.85
    xdim = ydim   * 1; * 0.85
  ENDELSE


  ;Make a little progress window
  startup = Widget_base(Column=1, /Base_Align_Center)
  temp = Widget_label(startup, Value='We are building your Polyviewer
Universe.')
  temp = Widget_label(startup, Value='Please be patient, building a
universe is hard work.')
  progress = Widget_label(startup, Value='', /Dynamic)
  temp = Widget_label(startup, Value='')
  temp = Widget_label(startup, Value='')

  Widget_control, startup, /REALIZE, xoffset=xdim/2, yoffset=ydim/2

  ;XMANAGER, '', startup, /no_block



  szsuperV = Size(superV)
```

```
  ; set up the plist. We do a lot of checking
  ;in case the users does something strange
  ; and to take care of legacy issues

  CASE szsuperV[1] OF
    4:BEGIN
    IF N_elements(plist2) EQ 0 THEN BEGIN
      pList2 = superP[0,Uniq(superP[0,*])]
      plist2 = Reform(plist2, N_elements(plist2))
    ENDIF
  END
  3:BEGIN
  IF (N_elements(plist2) EQ 0) AND (N_elements(vpLookup[*,0]) GE 3)
THEN BEGIN
    pList2 = vpLookup[0,*]
    plist2 = Reform(plist2, N_elements(plist2))
  ENDIF ELSE BEGIN
    IF (N_elements(plist2) EQ 0) THEN plist2=1
  ENDELSE
END
ENDCASE
;using plist instead of plist2 prevents polyviewer from changing
plist2
plist = plist2

;start setting up the custom naming of regions
IF N_elements(object_label2) EQ 0 THEN object_label2={ID:plist,
name:Strtrim(String(Long(plist)),2)}

nPart = N_elements(plist)
;Intiallize some of the optional array if they have not been done
so...
IF N_elements(Euler) EQ 0 THEN Euler = [-12.,-12.,-12.]
IF N_elements(Euler) EQ 3 THEN Euler = [Reform(plist, 1, nPart),
Rebin(Euler, 3, nPart, /sam)]

IF N_elements(Object_data) EQ 0 THEN Object_data = 0.
IF N_elements(Object_data) EQ 1 THEN Object_data = [Reform(plist, 1,
nPart), Reform(plist, 1, nPart)]

;stop
IF N_elements(Vertex_data) EQ 0 THEN Vertex_data = 0.
IF N_elements(Vertex_data) EQ 1 THEN Vertex_data =
(superV[szsuperV[1]-1,*])
szVD = Size(vertex_data)
If szVD[0] eq 1 then Vertex_data = Transpose(Vertex_data)
szVD = Size(vertex_data)

If N_elements(vertex_colors) eq 0 then vertex_colors=reform([0],1,1)

szVertexColors = Size(vertex_colors)

If szVertexColors[0] eq 1 Then vertex_colors =
Transpose(vertex_colors)
```

```
szVertexColors = Size(vertex_colors)

;
;stop
;First set up some ledgends and scroll bars:
; Create models.
oOIMLedgend = Obj_new('IDLgrModel', Hide=1) ; Will hold the oim unit
triangle
oColorLedgend  =  Obj_new('IDLgrModel',  Hide=0)  ;  will  hold  the
colorbar.
oLedgend = Obj_new('IDLgrModel'); will hold the layer for both legends
;Now set up the OIM ledgend.

Makeunittri2, image, /alpha

oUnitTri = Obj_new('IDLgrImage', image,$
  name="Unit     Triangle",     Interleave=0,     Location=[.45,-.9,0],
Dimension=[0.47, 0.47], Blend_Function=[3,4] )
text = Strarr(3)
text[0] = '(001)'
text[1] = '(101)'
text[2] = '(111)'


oFont = Obj_new('IDLgrFont', 'Helvetica', Size= Fix(xdim/35.))
oLedgentext   =   Obj_new('IDLgrText',   text,   Locations=([[0.4,   -
.95,0],[0.85,-0.95,0], [0.80, -0.42,0]]), $
  Color=[255,255,255], Font=oFont, Alpha_Channel=1.0)
oFont = Obj_new('IDLgrFont', 'Helvetica', Size= Fix(xdim/50.))
;oOrientationText  =  OBJ_NEW('IDLgrText',  ['Color  represents  crystal
direction parallel with [0,0,1]'] , $
;               Color   =   [255,255,255],   Locations=[-0.9,   .9,0]   ,
Font=oFont)
oOIMLEDGEND->Add, oUnitTri
oOIMLEDGEND->Add, oLedgentext
;oOIMLEDGEND->Add, oOrientationText
oLedgend->Add, oOIMLedgend


;Load a palette for coloring polygons/objects...this will only be used
for vertex_coloring and
; user_property coloring
oPalette = Obj_new('IDLgrPalette')
oPalette -> Loadct, 33 ; load up a default color palette

;set an inital range for the colorbar
mx = 1.0
mn = 0.0

xs = [-0.75, 1.]
ys = [0.89, 1]
oColorbar  =  Obj_new('IDLgrColorbar',  Dimensions  =  [1.5,  0.05],
ycoord_conv=ys, xcoord_conv=xs, show_outline=1)
oColorbar->Setproperty, Palette = oPalette
oCOLORLEDGEND->Add, oColorbar
```

```
size1 = Obj_new('IDLgrFont', 'Helvetica', Size=16)
xs = [-0.75-mn*1.5/(mx-mn) , 1.5/(mx-mn)]
oCAxis   =   Obj_new('IDLgrAxis',   0,   Color=[0,0,0],   Range=[mn,mx],
Location=[0,0.89 ], ticklen = 0.05, xcoord_conv=xs, /exact, minor=0)
oCAxis->Setproperty,              Tickvalues=Findgen(7)/6.*(mx-mn)+mn,
/Use_text_color
oCAxis->Getproperty, TickText=oAxisText
oAxisText->Setproperty,       Font=size1,       Recompute_dimensions=2,
Color=[127,127,127]
oCOLORLEDGEND->Add, oCAxis
oColorLedgend->Setproperty, Hide=1
oLedgend->Add, oColorLedgend

;and object array that will hold all the objects that will be part of
the microstructure
;this  includes  additional  planes,  and/or  points  added  to  the
visualization.
oPart = Objarr(nPart)


;Read  in  the  Polygon  and  vertex  lists  again,  there  is  a  lot  of
redunancies here because of legacy data
;formats,  and  checking  that  data  is  inputed  correctly.  Its  not
foolproof, as I have already proven to
;myself.

CASE szsuperV[1]  OF
  4:BEGIN
  Widget_control,  progress,  Set_Value='Reformatting  your  Polygon
List',/Dynamic
  ; this will reformat my old system of superV and superP, where the
vertex lists and polygon lists of all
  ; the regions were just concatenated into two lists, a superV and
superP, which had a leading column
  ; that indicated the region number.
  superVtemp = superV
  superPtemp = superP
  Supermeshvplookup, superVtemp, superPtemp, vptemp
  ; the new Supervtemp and superptemp list are now a consistent v and
p list.  The vptemp list tells you where to
  ; grab  the  polygons  and  vertexes  for  a  particular  region.   See
mesher.pro for more documentation.


  FOR i=0, nPart-1 DO BEGIN
    Widget_control,     progress,     Set_Value='Preparing     object:
'+Strcompress(i+1)+' of '+Strcompress(nPart),/Dynamic
    whp = Where(vptemp[0,*] EQ pList[i])
    whp = whp[0]
    IF (whp[0] NE -1) THEN BEGIN
      IF (vptemp[2,whp]-vptemp[1,whp] GE 3) THEN BEGIN
        vtemp2 = SuperVtemp[1:3,vptemp[3,whp]:vptemp[4,whp]]
        IF szVD[szVD[0]] EQ szV[szV[0]] THEN $
          IF (szVD[0] GT 1) THEN $
```

```
          vertex_datatemp  =  vertex_data[*,vptemp[3,whp]:vptemp[4,whp]]
$
        ELSE                        vertex_datatemp                        =
vertex_data[vptemp[3,whp]:vptemp[4,whp]] $
      ELSE vertex_datatemp = -1

      ptemp2 = SuperPtemp[1, vptemp[1,whp]:vptemp[2,whp] ]
      norms = Compute_mesh_normals(vtemp2, ptemp2)

      wh = Where(object_label2.id EQ plist[i])
      IF wh[0] GE 0 THEN name = object_label2.name[wh[0]] $
      ELSE name = Strtrim(String(Fix(plist[i])), 2)


      wh = Where(object_data[0,*] EQ plist[i])
      IF wh[0] GE 0 THEN odatatemp = Reform(object_data[1:*,wh[0]]) $
      ELSE odatatemp = plist[i]

      wh = Where(Euler[0,*] EQ plist[i])
      IF wh[0] GE 0 THEN eulertemp = euler[1:3,wh[0]] $
      ELSE eulertemp = [0.,0.,0.]


      oPart[i]        =        Obj_new('DJRgrPolygon',        vtemp2,
POLYGONS=Temporary(ptemp2), $
        SHADING=1,  NAME=name,  NORMALS=norms, $
        hidden_lines=1,Shininess=128,          Specular=[127,127,127],
Palette=oPalette, reject=0, $
        regionid=plist[i])

      wh = Where(object_data[0,*] EQ plist[i])
      IF wh[0] GE 0 THEN $
        oPart[i]->Setproperty,              object_data              =
Reform(object_data[1:*,wh[0]])


      wh = Where(Euler[0,*] EQ plist[i])
      IF wh[0] GE 0 THEN oPart[i]->Setproperty, euler=euler[1:3,wh[0]]

      IF szVD[szVD[0]] EQ szV[szV[0]] THEN $
        IF (szVD[0] GT 1) THEN $
        vertex_datatemp = vertex_data[*,vptemp[3,whp]:vptemp[4,whp]] $
      ELSE  vertex_datatemp  =  vertex_data[vptemp[3,whp]:vptemp[4,whp]]
$
    ELSE vertex_datatemp = -1

    oPart[i]->Setproperty, vertex_data = vertex_datatemp


  ENDIF ELSE BEGIN
    ;only two vertices defined, which does not make a great polygon,
it will not be
    ;added to the visualization.
    name = Strcompress(String(Fix(plist[i])), /remove)
    oPart[i] = Obj_new(NAME=name)
```

```
  ENDELSE
ENDIF ELSE BEGIN
  ; the region was not found in the vp list
  ;skip it
  name = Strcompress(String(Fix(plist[i])), /remove)
  oPart[i] = Obj_new(NAME=name)
ENDELSE
ENDFOR
END
3:BEGIN
CASE N_elements(vpLookup[*,0]) OF
  3: BEGIN
    FOR i=0, nPart-1 DO BEGIN
      Widget_control,    progress,    Set_Value='Preparing    object:
'+Strcompress(i+1)+' of '+Strcompress(nPart),/Dynamic
      whp = Where(vpLookup[0,*] EQ pList[i])
      whp = whp[0]
      IF (whp[0] NE -1) THEN BEGIN
        IF (vpLookup[2,whp]-vpLookup[1,whp] GE 3) THEN BEGIN
          ;ptemp = superP[ vpLookup[1,whp]:vpLookup[2,whp] ]
          ;vtemp = superV
          ;this removes all the extra vertices that are not referenced
in ptemp
          ;trash = Mesh_validate(vtemp, ptemp, /pack)
          ntri         =         djr_pack_verts(superV,         superP[
vpLookup[1,whp]:vpLookup[2,whp] ],  $
              ;vtemp,    ptemp,    aux_data_in    =    vertex_data,
aux_data_out=vertex_data_temp)
              vtemp, ptemp, v_index=v_intemp)
          ;this   disables   the   ability   to   affectively   use
vertex_data....
          IF ntri GE 0 THEN BEGIN
            norms = Compute_mesh_normals(vtemp, ptemp)
            wh = Where(object_label2.id EQ plist[i])
            IF wh[0] GE 0 THEN name = object_label2.name[wh[0]] $
            ELSE name = Strtrim(String(Fix(plist[i])), 2)


            oPart[i]       =       Obj_new('DJRgrPolygon',       vtemp,
POLYGONS=Temporary(ptemp), $
              SHADING=1 ,NAME=name,   NORMALS=norms, depth_offset = 0,
$
              hidden_lines=1,Shininess=128,    Specular=[127,127,127],
Palette=oPalette, reject = 0, $
              regionid=plist[i])


            wh = Where(object_data[0,*] EQ plist[i])
            IF wh[0] GE 0 THEN $
              oPart[i]->Setproperty,           object_data           =
Reform(object_data[1:*,wh[0]])

              If  n_elements(v_intemp)  eq  0  then  vertex_data_temp=-1
else $
                vertex_data_temp = vertex_data[*,v_intemp]
```

```
              ;stop
             oPart[i]->Setproperty, vertex_data = vertex_data_temp

            If szVertexColors[2] eq szV[2] THEN $
              vertex_colors_temp = vertex_colors[*, v_intemp]

            oPart[i]->Setproperty, vert_colors = vertex_colors_temp


            wh = Where(Euler[0,*] EQ plist[i])
            IF    wh[0]    GE    0    THEN    oPart[i]->Setproperty,
euler=euler[1:3,wh[0]]
          ENDIF ELSE BEGIN
            name = Strcompress(String(Fix(plist[i])), /remove)
            oPart[i] = Obj_new(NAME=name)
          ENDELSE
        ENDIF ELSE BEGIN
          ; no polygons found
          name = Strcompress(String(Fix(plist[i])), /remove)
          oPart[i] = Obj_new(NAME=name)
        ENDELSE
      ENDIF ELSE BEGIN
        ;no region found
        name = Strcompress(String(Fix(plist[i])), /remove)
        oPart[i] = Obj_new(NAME=name)
      ENDELSE
    ENDFOR


  END
  5: BEGIN ; this is most common state of the vp list for me.
    FOR i=0, nPart-1 DO BEGIN
      Widget_control,    progress,    Set_Value='Preparing    object:
'+Strcompress(i+1)+' of '+Strcompress(nPart),/Dynamic
      whp = Where(vpLookup[0,*] EQ pList[i])
      whp = whp[0]
      IF (whp[0] NE -1) THEN BEGIN
        IF (vpLookup[2,whp]-vpLookup[1,whp] GE 0) THEN BEGIN
          ptemp = superP[ vpLookup[1,whp]:vpLookup[2,whp] ]
          vtemp = superV[0:2, vpLookup[3,whp]:vpLookup[4,whp] ]



          nPoly = N_elements(ptemp)
          vbegin = vpLookup[3,whp]

          IF N_elements(Uniq(ptemp[0:*:4])) NE 1 THEN BEGIN ; Check to
make sure that poly only has triangles
            k = 0
            WHILE k LT nPoly DO BEGIN
              ptemp[k+1:k+ptemp[k]] = ptemp[k+1:k+ptemp[k]]-vbegin[0]
              k = k+ptemp[k]+1
            ENDWHILE
          ENDIF ELSE BEGIN ; If only triangles, things go much easier
```

```
            ptemp = Reform(Temporary(ptemp), 4,nPoly/4)
            ptemp[1:3,*] -= vbegin[0]
            ptemp = Reform(Temporary(ptemp), nPoly)
          ENDELSE
          ;norms = Compute_mesh_normals(vtemp, ptemp)

          wh = Where(object_label2.id EQ plist[i])
          IF wh[0] GE 0 THEN name = object_label2.name[wh[0]] $
          ELSE name = Strtrim(String(Fix(plist[i])), 2)

          oPart[i]        =        Obj_new('DJRgrPolygon',        vtemp,
POLYGONS=Temporary(ptemp), $
            SHADING=1,  NAME=name,$;  NORMALS=norms, $
            hidden_lines=1,Shininess=128,       Specular=[127,127,127],
REJECT=0, $
            Palette=oPalette, regionid=plist[i])

          wh = Where(object_data[0,*] EQ plist[i])
          IF wh[0] GE 0 THEN $
            oPart[i]->Setproperty,             object_data             =
Reform(object_data[1:*,wh[0]])


          wh = Where(Round(Euler[0,*]) EQ plist[i])
          IF     wh[0]     GE     0     THEN     oPart[i]->Setproperty,
euler=euler[1:3,wh[0]]

          IF szVD[szVD[0]] EQ szV[szV[0]] THEN $
            IF (szVD[0] GT 1) THEN $
            vertex_datatemp                                             =
vertex_data[*,vplookup[3,whp]:vplookup[4,whp]] $
          ELSE                    vertex_datatemp                       =
vertex_data[vplookup[3,whp]:vplookup[4,whp]]$
        ELSE vertex_datatemp=-1


        oPart[i]->Setproperty, vertex_data = vertex_datatemp

        If szVertexColors[2] eq szV[2] THEN $
          vertex_colors_temp             =             vertex_colors[*,
vplookup[3,whp]:vplookup[4,whp]]

        oPart[i]->Setproperty, vert_colors = vertex_colors_temp


      ENDIF ELSE BEGIN
        ; no polygons found
        name = Strcompress(String(Fix(plist[i])), /remove)
        oPart[i] = Obj_new(NAME=name)
      ENDELSE
    ENDIF ELSE BEGIN
      ;no region found
      name = Strcompress(String(Fix(plist[i])), /remove)
      oPart[i] = Obj_new(NAME=name)
    ENDELSE
```

```
  ENDFOR


END
ELSE:BEGIN
;; no vp list found, or properly formatted, just load superV and
superP into one object
;
;;norms       =       Compute_Mesh_Normals(superV,       REFORM(superP,
N_ELEMENTS(superP)))
;
;If szVD[szVD[0]] eq szV[szV[0]] THEN $
;          vertex_datatemp = vertex_data $
;ELSE vertex_datatemp = -1
;
;wh = where(object_label2.id eq plist[i])
;If wh[0] ge 0 then name = object_label2.name[wh[0]] $
;     ELSE name = STRTRIM(String(Fix(plist[i])), 2)
;
;wh = where(object_data[0,*] eq plist[i])
;                If     wh[0]     ge     0     then     odatatemp     =
Reform(object_data[1:*,wh[0]]) $
;ELSE odatatemp = plist[i]
;
;wh = where(Euler[0,*] eq plist[i])
;                If wh[0] ge 0 then eulertemp = euler[1:3,wh[0]] $
;          ELSE eulertemp = [0.,0.,0.]
;
;
;oPart[0] = OBJ_NEW('DJRgrPolygon', superV, POLYGONS=superP, $
;     SHADING=1,         name=name,     REJECT=1     ,     Hidden_lines=1,
Specular=[127,127,127], Palette=oPalette, $
;     vertex_data    =    vertex_datatemp,    Object_data    =    odatatemp,
Euler=eulertemp, regionid=plist[i])
;;If Keyword_set(vert_colors) then BEGIN
;;    oPart[0]->SetProperty, Vert_Colors= vert_colors
;;ENDIF
ENDELSE
ENDCASE
END
ELSE: BEGIN ; nothing found...
  Print, "The vertex or polygon list is improperly formatted"
  Widget_control, startup, /destroy
  Return
END
ENDCASE
;clear out some memory...
superVtemp=0
superPtemp=0
vtemp = 0
ptemp = 0
ptemp2 = 0
vrtex_datatemp = 0

; clear out any regions that were not found
```

```
wh = Where(oPart NE Obj_new())
IF wh[0] EQ -1 THEN BEGIN
  Print, "No Requested objects found"
  Widget_control, startup, /destroy
  Return
ENDIF ELSE BEGIN
  plist  = plist[wh]
  oPart = oPart[wh]
  nPart = N_elements(oPart)
ENDELSE
;keep a copy of the plist that is only the regions, not planes or
points...
plist3 = plist

pinfo = Strarr(nPart)

FOR i=0, nPart-1 DO BEGIN
  pinfo[i] = String(pList[i])
ENDFOR

currentObj = oPart

plist = String(Fix(plist))
plist = Reform(plist, N_elements(plist))


;Set up some visualiztion stuff
;Colors->defualt [200,200,211] nice steel gray

dColor = [200,200,255]


;check and see if uColor is defined.  If not the default color will be
used.
;otherwise, by default, the user color will be used.

;user colors can support rgb or rgba formatting.
IF Keyword_set(uColor) THEN BEGIN
  temp = Intarr(5,nPart)+255
  temp[0,*] = plist
  CASE N_elements(uColor) OF
    3:BEGIN
    FOR i=0, npart-1 DO temp[1:3,i] = uColor
  END
  4:BEGIN
  FOR i=0, npart-1 DO temp[1:4,i] = uColor
END
ELSE:BEGIN
FOR i=0, npart-1 DO temp[1:3,i] = dColor
IF N_elements(uColor[*,0]) EQ 5 THEN alpha = uColor[4,*] ELSE alpha =
Bytarr(N_elements(uColor[0,*]) > 1)+255
FOR i=0, npart-1 DO BEGIN
  wh = Where(uColor[0,*] EQ temp[0,i])
  IF wh[0] EQ -1 THEN temp[1:4,i] = [dColor,255] ELSE temp[1:4,i] =
[uColor[1:3,wh],alpha[wh]]
```

```
ENDFOR
END
ENDCASE
uColor = temp

ENDIF ELSE BEGIN

  uColor = Intarr(5,nPart)+255
  uColor[0,*] = plist
  FOR i=0, npart-1 DO uColor[1:3,i] = dColor
ENDELSE

uColorPt = Ptr_new(uColor)

FOR i=0, nPart-1 DO BEGIN
 oPart[i]->Setproperty,                              Color=uColor[1:3,i],
Alpha_Channel=(Float(uColor[4,i])/255. > 0) < 1.0
ENDFOR

;Check the object_data ranges, and the vertex_data ranges, and save a
copy.
od_ncolumns = N_elements(object_data[*,0])-1

objDataRange=Fltarr(2,od_ncolumns)
IF szVD[0] EQ 1 THEN vd_ncolumns=1 ELSE $
  vd_ncolumns = szvd[1]
vertDataRange=Fltarr(2,vd_ncolumns)

objDataRange[0,*]  = !values.f_infinity
objDataRange[1,*]  = -!values.f_infinity
vertDataRange[0,*]  = !values.f_infinity
vertDataRange[1,*]  = -!values.f_infinity

FOR i=0, nPart-1 DO BEGIN
  FOR j=0, od_ncolumns-1 DO BEGIN
    ;oPart[i]->SetProperty, od_column=j
    temp = oPart[i]->Get_current_obj_data(j)

    IF Finite(temp) GE 1 THEN BEGIN
      objDataRange[0,j] = objDataRange[0,j] < temp
      objDataRange[1,j] = objDataRange[1,j] > temp
    ENDIF
  ENDFOR
  oPart[i]->Setproperty, vd_column=0

  FOR j=0, vd_ncolumns-1 DO BEGIN
    oPart[i]->Setproperty, vd_column=j
    temp = oPart[i]->Get_vert_data_range()
    IF N_elements(temp) GT 1 THEN BEGIN
      vertDataRange[0,j] = vertDataRange[0,j] < temp[0]
      vertDataRange[1,j] = vertDataRange[1,j] > temp[1]
    ENDIF
  ENDFOR
  oPart[i]->Setproperty, vd_column=0
```

```
ENDFOR

IF objDataRange[0] GE objDataRange[1] THEN objDataRange[0] =
objDataRange[1]-1

;Now Figure out the data ranges
xrange = [1e9,-1.0]
yrange = [1e9,-1.0]
zrange = [1e9,-1.0]

FOR i=0, nPart-1 DO BEGIN
  oPart[i]->Getproperty, XRANGE=xrangetemp, YRANGE=yrangetemp,
ZRANGE=zrangetemp
  xrange[0] = xrange[0] < xrangetemp[0]
  yrange[0] = yrange[0] < yrangetemp[0]
  zrange[0] = zrange[0] < zrangetemp[0]

  xrange[1] = xrange[1] > xrangetemp[1]
  yrange[1] = yrange[1] > yrangetemp[1]
  zrange[1] = zrange[1] > zrangetemp[1]
ENDFOR
; check for zero ranges.
IF xrange[0] EQ xrange[1] THEN xrange[1] += 1.e-6
IF yrange[0] EQ yrange[1] THEN yrange[1] += 1.e-6
IF zrange[0] EQ zrange[1] THEN zrange[1] += 1.e-6
xmin = xrange[0]
xmax = xrange[1]
ymin = yrange[0]
ymax = yrange[1]
zmin = zrange[0]
zmax = zrange[1]



;Add in the optional planes.

IF Keyword_set(PLANES) THEN BEGIN

  ;make the vetex/polygons for the arrows
  arrowpoints = [[0,0,0], [0,0,.05], [0,0.915, 0.05], [0,0.8,0.2],
[0,1.,0]]
  arrowpoints[1,*] = arrowpoints[1,*] /2.0
  arrowpoints[2,*] = arrowpoints[2,*] /8.
  Mesh_obj, 6, arrowV, arrowP,arrowpoints, /closed, p1 = 15, p3 =
[0,1,0]
  FOR aI =0, N_elements(arrowV[0,*])-1 DO arrowV[*,aI] =
arrowV[*,aI]*[2, 0.75, 2]
  vTemp = arrowP
  arScale = Max([xrange[1]-xrange[0],yrange[1]-yrange[0], zrange[1]-
zrange[0] ])

  ;_____________________________
  ;object for holding the planes

  plname=0
```

```
  FOR i=0, N_elements(planes[0,*])-1 DO BEGIN
    plane = planes[*,i]
    arrowang = Acos(Transpose(plane[0:2])#[0,1,0])
    arrowaxis = Crossp(plane[0:2], [0,1,0])
    Misvect2rotmat, rotarrow, arrowang, arrowaxis
    vTemp = arrowV
    arScale = Max([xrange[1]-xrange[0],yrange[1]-yrange[0], zrange[1]-
zrange[0] ])


    srt = Reverse(Sort(Abs(plane[0:2])))
    CASE srt[0] OF
      2: BEGIN
        planeV = [[xmin, ymin, 0.], [xmax, ymin, 0], [xmax, ymax,0],
[xmin, ymax, 0]]
        planeV[2,*]                                                    =
(plane[0]*planeV[0,*]+plane[1]*planeV[1,*]+plane[3])/(-
plane[2]+0.000000001)

        FOR aI = 0, N_elements(vTemp[0,*])-1 DO BEGIN
          vTemp[0:2,aI] = rotarrow##vTemp[0:2,aI]
          vTemp[0:2,aI] = vTemp[0:2,aI] * arScale
          vTemp[0:2,aI] = vTemp[0:2,aI] + [(xmax-xmin)/2.0+xmin,(ymax-
ymin)/2.0 +ymin,$
            (plane[0]*((xmax-xmin)/2.0+xmin)+plane[1]*((ymax-
ymin)/2.0+ymin)+plane[3])/(-plane[2]+0.000000001)  ]

        ENDFOR


      END
      1: BEGIN
        planeV = [[xmin, 0., zmin], [xmax, 0, zmin], [xmax, 0,zmax],
[xmin, 0, zmax]]
        planeV[1,*]                                                    =
(plane[0]*planeV[0,*]+plane[2]*planeV[2,*]+plane[3])/(-
plane[1]+0.000000001)

        FOR aI = 0, N_elements(vTemp[0,*])-1 DO BEGIN
          vTemp[0:2,aI] = rotarrow##vTemp[0:2,aI]
          vTemp[0:2,aI] = vTemp[0:2,aI] * arScale
          vTemp[0:2,aI]      =      vTemp[0:2,aI]      +      [(xmax-
xmin)/2.0+xmin,(plane[0]*((xmax-xmin)/2.0+xmin) $
            +plane[2]*((zmax-zmin)/2.0+zmin)+plane[3])/(-
plane[1]+0.000000001),$
            (zmax-zmin)/2.0 + zmin ]

        ENDFOR


      END
      0: BEGIN
        planeV = [[0., ymin, zmin], [0, ymax, zmin], [0, ymax,zmax],
[0, ymin, zmax]]
```

```
        planeV[0,*]                                                    =
(plane[1]*planeV[1,*]+plane[2]*planeV[2,*]+plane[3])/(-
plane[0]+0.000000001)

        FOR aI = 0, N_elements(vTemp[0,*])-1 DO BEGIN
          vTemp[0:2,aI] = rotarrow##vTemp[0:2,aI]
          vTemp[0:2,aI] = vTemp[0:2,aI] * arScale
          vTemp[0:2,aI]    =    vTemp[0:2,aI]    +    [(plane[2]*((zmax-
zmin)/2.0+zmin)+plane[1]*((ymax-ymin)/2.0+ymin)$
            +plane[3])/(-plane[0]+0.000000001),(ymax-ymin)/2.0 +ymin,$
            (zmax-zmin)/2.0+zmin ]

        ENDFOR


      END
    ENDCASE

    name = Strcompress('Plane' + String(plname), /remove_all)
    oPart = [oPart, Obj_new('IDLgrPolygon', planeV, SHADING=1, Color =
[255,25,25], NAME=name, specular=[127,127,127], alpha=0.5 )]
    plist=[plist,name]
    ;name = STRCOMPRESS('Arrow' + String(plname), /remove_all)
    ;oPart = [oPart, OBJ_NEW('IDLgrPolygon', vTemp,Polygons = arrowP,
SHADING=1, Color = [255,25,25], NAME = name, specular=[127,127,127])]
    ;plist=[plist,name]
    plname = plname+1


  ENDFOR
ENDIF
;___________________



;add in x,y,z points? Sure why not
IF Keyword_set(points) THEN BEGIN
  Mesh_obj, 4, vtemp, ptemp, Replicate(1,8,8), /closed
  sphere    =    Obj_new('IDLgrPolygon',    vtemp,    Polygons=ptemp,
color=[255,0,0])
  psym = Obj_new('IDLgrSymbol', Data = 3, size = 1 )
  ;psym = OBJ_NEW('IDLgrSymbol', Data = sphere, size = 1)

  oPart    =    [oPart,Obj_new('IDLgrPolyLine',    points,    Shading=1,
Name='Points', Symbol=psym, linestyle=6, Color=[255,0,0])]
  plist = [plist, 'points']

ENDIF

;I'm done creating objects, get a list of all the names
object_label = Strarr(N_elements(oPart))
FOR i=0, N_elements(oPart)-1 DO BEGIN
  oPart[i]->Getproperty, Name = name
  object_label[i] =  name
ENDFOR
```

```
;For i=0, N_ELEMENTS(oPart)-1 Do Begin
;     wh       =       where(STRCOMPRESS(String(Fix(Object_Label2.ID)),
/remove_all) EQ object_label[i])
;     IF wh[0] NE -1 THEN object_label[i] = Object_Label2.Name[wh[0]]
;ENDFOR



;Now its time to set up all of the viewing space
; Compute viewplane rect based on aspect ratio.
aspect = Float(xdim) / Float(ydim)

; get the view for the initial zoom
IF N_elements(initialview) GE 4 THEN $
  zoomval = initialview[3] $
ELSE zoomval = 50 ;  goes from 0 to 100
zoom = Strcompress( String(zoomVal) , /remove_all )
Getview, zoom, aspect, myView

; Create view.
oView = Obj_new('IDLgrView', PROJECTION=2, EYE=2, ZCLIP=[1.5,-1.5],$
  VIEWPLANE_RECT=myview,       COLOR=[0,0,0],       name='MICROVIEW');,
Depth_Cue=[-0.5, 1.5],)
oViewOIM = Obj_new('IDLgrView', Transparent=1, name='LEGENDVIEW')
oViewGroup = Obj_new('IDLgrViewGroup')

oViewGroup->Add, oView
oViewGroup->Add, oViewOIM


oTop = Obj_new('IDLgrModel')
oGroup = Obj_new('IDLgrModel')
oMicro = Obj_new('IDLgrModel')
oXstal = Obj_new('IDLgrModel')
oNoAxis = Obj_new('IDLgrModel')
oTop->Add, oGroup
oGroup-> Add, oMicro
oMicro->Add, oNoAxis

;Set up the Microstrucutre model
; Compute data bounds.
xmm = xMax - xMin
ymm = yMax - yMin
zmm = zMax - zMin
xyzmm = [xmm, ymm, zmm ]

; Compute coordinate conversion to normalize.
xyzSpan = Max( xyzmm )
xs = [0.0,1.0/xyzSpan]
ys = [0.0,1.0/xyzSpan]
zs = [0.0,1.0/xyzSpan]

xs[1] = Min( [xs[1], ys[1], zs[1]] ) *0.85
ys[1] = xs[1]
```

```
zs[1] = ys[1]

xs[0] = -0.5*(xMax+xMin)*xs[1]
ys[0] = -0.5*(yMax+yMin)*ys[1]
zs[0] = -0.5*(zMax+zMin)*zs[1]


; Set up the microstructure axis
xrange=[ xMin, xMax ]
yrange=[ yMin, yMax ]
zrange=[ zMin, zMax > (zmin+0.01*(xmax-xmin)) ]


size1 = Obj_new('IDLgrFont', 'Helvetica', Size=18)
size2 = Obj_new('IDLgrFont', 'Helvetica', Size=24)

xTitle   =   Obj_new('IDLgrText',   'X   [   !Mmm   ]',   FONT=size2,
Color=[255,255,0], Recompute_Dimensions=2, Enable_Format=1)
yTitle   =   Obj_new('IDLgrText',   'Y   [   !Mmm   ]',   FONT=size2,
Color=[255,255,0], Recompute_Dimensions=2, Enable_Format=1)
zTitle   =   Obj_new('IDLgrText',   'Z   [   !Mmm   ]',   FONT=size2,
Color=[255,255,0], Recompute_Dimensions=2, Enable_Format=1)

xAxis = Obj_new("IDLgrAxis", 0, Color=[0,255,0], Ticklen=0.02/xs[1], $
  TITLE=xTitle,   /EXACT,   RANGE=xrange-xmin,   Thick   =   2.0   ,
Location=[xmin, yMin, zMin], Name='X AXIS')
xAxis->Getproperty, TickText=xAxisText
xAxisText->Setproperty, Font=size1, Recompute_Dimensions=2
xAxis->Getproperty, Xcoord_Conv=xcoord
xcoord[0] = xcoord[0]+xmin
xAxis->Setproperty, Xcoord_Conv=xcoord


yAxis = Obj_new("IDLgrAxis", 1, Color=[0,255,0], Ticklen=0.02/xs[1], $
  Range=yrange-ymin,    /EXACT,    TITLE=yTitle,    Thick    =    2.0,
Location=[xMin, yMin, zMin], Name='Y Axis')
yAxis->Getproperty, TickText=yAxisText
yAxisText->Setproperty, Font=size1, Recompute_Dimensions=2
yAxis->Getproperty, ycoord_Conv=ycoord
ycoord[0] = ycoord[0]+ymin
yAxis->Setproperty, ycoord_Conv=ycoord



zAxis = Obj_new("IDLgrAxis", 2, Color=[0,255,0], Ticklen=0.02/xs[1], $
  Range=zrange-zmin,    /EXACT,    TITLE=zTitle,    Thick    =    2.0,
Location=[xMin, ymax, zMin], Name='Z Axis')
zAxis->Getproperty, Zcoord_Conv=zcoord
zcoord[0] = zcoord[0]+zmin
zAxis->Setproperty, Zcoord_Conv=zcoord
;zAxis->SetProperty, Textbaseline=[0,1,0], tickdir=0
;zTitle->SetProperty, Baseline=[0,1,0], updir=[0,0,1], alignment = 1

zAxis->Getproperty, TickText=zAxisText
zAxisText->Setproperty, Font=size1, Recompute_Dimensions=2
```

```
oMicro-> Add, xAxis
oMicro-> Add, yAxis
oMicro-> Add, zAxis
oNoAxis->Add, oPart
;ians no axis code...
IF Keyword_set(noaxis) THEN BEGIN
  xAxis->Setproperty, hide=1
  yAxis->Setproperty, hide=1
  zAxis->Setproperty, hide=1
ENDIF


;Now scale the microstructure so that it fits in the viewing box.
zoomval = xs[1]
oMicro->Scale,     xs[1],ys[1],zs[1]
oMicro->Translate, xs[0],ys[0],zs[0]

;Set up the clip planes for bottom and top clipping

ClipPlanes  =  [  [0,0,1,-(zMax-Abs(0.025*(zMax-zMin)))  ]  ,  [0,0,-
1,(zMin+Abs(0.025*(zMax-zMin))) ] ]

; Rotate to standard view for first draw.
oGroup->Rotate, [1,0,0], -90     ; -40
oGroup->Rotate, [0,1,0], -45     ; 20
oGroup->Rotate, [1,0,0], 30; 0

oGroup->Getproperty, Transform = origTransform

IF N_elements(initialview) GE 3 THEN BEGIN
  Setypr, origTransform, initialview[0:2]
  oGroup->Setproperty, Transform = origTransform
ENDIF


;Set up the non-rotation oTop model.  Even if lights say rotating,
they don't
; Create some lights.  Making two lighting models, both non-rotating
oL1 = Obj_new('IDLgrModel')
oL2 = Obj_new('IDLgrModel')
oLOIM = Obj_new('IDLgrModel') ;OIM lighting model
oTop->Add, oL1
oTop->Add, oL2
oTop->Add, oLOIM


oLights1   =   Obj_new('IDLgrLight',   Location=[-2,2,10],   TYPE=2,
INTENSITY=0.6)
oLights2   =   Obj_new('IDLgrLight',   Location=[1,-2,-6],   TYPE=2,
INTENSITY=0.5)
oLights3 = Obj_new('IDLgrLight', TYPE=0, INTENSITY=0.25)

oL1->Add, oLights1
oL1->Add, oLights2
```

```
oL1->Add, oLights3


ambientLight = Obj_new('IDLgrLight', Type=0, Intensity=0.2)
nonrotatingLight1 = Obj_new('IDLgrLight', Type=2, Intensity=0.40, $
  Location=[xrange[1], yrange[1], 4*zrange[1]], $
  Direction=[xrange[0], yrange[0], zrange[0]])
fillLight = Obj_new('IDLgrLight', Type=3, Intensity=0.7, $
  Location=[(xrange[1]-xrange[0])/2.0,   (yrange[1]-yrange[0])/2.0,   -
2*Abs(zrange[0])], $
  Direction=[(xrange[1]-xrange[0])/2.0,      (yrange[1]-yrange[0])/2.0,
zrange[1]])
nonrotatingLight2 = Obj_new('IDLgrLight', Type=1, Intensity=0.3, $
  Location=[-xrange[1], (yrange[1]-yrange[0])/2.0, 4*zrange[1]], $
  Direction=[xrange[1], (yrange[1]-yrange[0])/2.0, zrange[0]])



oL2->Add, ambientLight
oL2->Add, fillLight
oL2->Add, nonrotatingLight1
oL2->Add, nonrotatinglight2


nonrotatingLight1->Setproperty,     XCoord_Conv=xs,     YCoord_Conv=ys,
ZCoord_Conv=zs
fillLight->Setproperty, XCoord_Conv=xs, YCoord_Conv=ys, ZCoord_Conv=zs
nonrotatingLight2->Setproperty,     XCoord_Conv=xs,     YCoord_Conv=ys,
ZCoord_Conv=zs

oOIML1    =    Obj_new('IDLgrLight',    Location=[-2,2,5],    TYPE=2,
INTENSITY=0.55)
oOIMl2    =    Obj_new('IDLgrLight',    Location=[2,2,5],    TYPE=2,
INTENSITY=0.55)
oOIMl3    =    Obj_new('IDLgrLight',    Location=[0,-3,-2],    TYPE=2,
INTENSITY=0.41)
oOIML4 = Obj_new('IDLgrLight', TYPE=0, INTENSITY=0.05)

oLOIM->Add, oOIML1
oLOIM->Add, oOIML2
oLOIM->Add, oOIML3
oLOIM->Add, oOIML4

oL1->Setproperty, Hide=0
oL2->Setproperty, Hide=1
oLOIM -> SetProperty, Hide=1



;Final setup of viewing container
; Place the top model in the view.
oView->Add, oTop
oViewOIM ->Add, oLEDGEND
; Create a trackball.
oTrack = Obj_new('Trackball', [xdim/2.0, ydim/2.0], xdim/2.0)
```

```
; Create a holder object for easy destruction.
oHolder = Obj_new('IDL_Container')
oHolder->Add, oViewGroup
oHolder->Add, oTrack




;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;
;Now Create the GUI

; Create the widgets.
wBase = Widget_base(/COLUMN, XPAD=0, YPAD=0, $
  TITLE="Polygon Viewer", /TLB_KILL_REQUEST_EVENTS, $
  TLB_FRAME_ATTR=1, MBAR=barBase, uname = 'MAINBASE')

;Create the menu bar.
; FILE Menu
fileMenu = Widget_button(barBase, VALUE='File', /MENU)

QuitButton = Widget_button(fileMenu, VALUE='Quit', UVALUE='QUIT', $
  UNAME='sState:quit')
; OUTPUT Menu
outputMenu = Widget_button(barBase, VALUE='Output', /MENU)

button = Widget_button(outputMenu, Value='TIFF File', $
  UValue='TIFF', Event_Pro='polyview_Output')
button = Widget_button(outputMenu, Value='JPEG File', $
  UValue='JPEG', Event_Pro='polyview_Output')
button = Widget_button(outputMenu, Value='EPS File', $
  UValue='EPS', Event_Pro='polyview_Output')
button = Widget_button(outputMenu, Value='Movie', $
  UValue='Movie', Event_Pro='polyview_Output')
button = Widget_button(outputMenu, Value='Movie in time', $
  UValue='MovTime', Event_Pro='polyview_Output')

; PROPERTIES Menu
propMenu = Widget_button(barBase, VALUE='Properties', /MENU)

; Background Color
bcolor = Widget_button(propMenu, Value='Background Color', /Menu)
dummy = Widget_button(bcolor, Value='Black', $
  Event_Pro='polyview_Properties', UValue='BBLACK')
dummy = Widget_button(bcolor, Value='White', $
  Event_Pro='polyview_Properties', UValue='BWHITE')
dummy = Widget_button(bcolor, Value='Charcoal', $
  Event_Pro='polyview_Properties', UValue='BCHARCOAL')
dummy = Widget_button(bcolor, Value='Gray', $
  Event_Pro='polyview_Properties', UValue='BGRAY')
```

```
; Axes Color
acolor = Widget_button(propMenu, Value='Axes Color', /Menu)
dummy = Widget_button(acolor, Value='Black', $
  Event_Pro='polyview_Properties', UValue='ABLACK')
dummy = Widget_button(acolor, Value='White', $
  Event_Pro='polyview_Properties', UValue='AWHITE')
dummy = Widget_button(acolor, Value='Yellow', $
  Event_Pro='polyview_Properties', UValue='AYELLOW')
dummy = Widget_button(acolor, Value='Green', $
  Event_Pro='polyview_Properties', UValue='AGREEN')
dummy = Widget_button(acolor, Value='Navy Blue', $
  Event_Pro='polyview_Properties', UValue='ANAVY')

; Drag Quality
dragButton = Widget_button(propMenu, VALUE="Drag Quality", /MENU)
dummy = Widget_button(DragButton, $
  VALUE='Really                Low',                UVALUE='VERYLOW',
Event_Pro='polyview_Properties')
dummy = Widget_button(DragButton, $
  VALUE='Low', UVALUE='LOW', Event_Pro='polyview_Properties')
dummy = Widget_button(DragButton, $
  VALUE='Medium', UVALUE='MEDIUM', Event_Pro='polyview_Properties')
dummy = Widget_button(DragButton, $
  VALUE='High', UVALUE='HIGH', Event_Pro='polyview_Properties')

; BackFace Culling
bfcButton = Widget_button(propMenu, VALUE="Backface Culling", /MENU)
dummy = Widget_button(bfcButton, $
  VALUE='Negative              (default)',              UVALUE='BFC',
UNAME='BFCNeg',Event_Pro='polyview_Properties')
dummy = Widget_button(bfcButton, $
  VALUE='Positive',                                     UVALUE='BFC',
UNAME='BFCPos',Event_Pro='polyview_Properties')
dummy = Widget_button(bfcButton, $
  VALUE='None         (slowest)',         UVALUE='BFC',UNAME='BFCNone',
Event_Pro='polyview_Properties')

;; Original View
;OGViewButton = Widget_Button(propMenu, VALUE="Reset", $
;     /SEPARATOR, UVALUE='OG', Event_Pro='polyview_Properties')


; LIGHTING Menu
helpMenu = Widget_button(barBase, VALUE='Lighting', /MENU)
button = Widget_button(helpMenu, Value='Scheme 1', $
  UValue='ONE', Event_Pro='Polyview_Properties')
button = Widget_button(helpMenu, Value='Scheme 2 (default)', $
  UValue='TWO', Event_Pro='Polyview_Properties')
button = Widget_button(helpMenu, Value='Scheme 3 (default)', $
  UValue='THREE', Event_Pro='Polyview_Properties')
; Coloring Menu
colorMenu = Widget_button(barBase, VALUE='Object Coloring', /MENU)
button = Widget_button(colorMenu, Value='Default Color', $
```

```
  UValue='defaultC', Event_Pro='DaveColor_Polygons')
button = Widget_button(colorMenu, Value='Random Color', $
  UValue='randomC', Event_Pro='DaveColor_Polygons')
button = Widget_button(colorMenu, Value='User Color', $
  UValue='userC', Event_Pro='DaveColor_Polygons')
button = Widget_button(colorMenu, Value='2D OIM Coloring', $
  UValue='2DOIM',                        Event_Pro='DaveColor_Polygons',
Sensitive=(N_elements(euler) GE 4))
button = Widget_button(colorMenu, Value='Crystal Normal Coloring', $
  UValue='normalC',                      Event_Pro='DaveColor_Polygons',
Sensitive=(N_elements(euler) GE 4))
button = Widget_button(colorMenu, Value='Normal Coloring', $
  UValue='normalC2', Event_Pro='DaveColor_Polygons')
button = Widget_button(colorMenu, Value='Object Property', $
  UValue='OBJECTPROPERTY', Event_Pro='DaveColor_Polygons', $
  Sensitive=1 )
;button = Widget_Button(colorMenu, Value='Points/Wire/Fill', $
;     UValue='wire', Event_Pro='DaveColor_Polygons')

; HELP Menu
helpMenu = Widget_button(barBase, VALUE='Help', /MENU)
button = Widget_button(helpMenu, Value='About', $
  UValue='ABOUT', Event_Pro='Polyview_Help')
button = Widget_button(helpMenu, Value='Help', $
  UValue='HELP', Event_Pro='Polyview_Help')

; Create a sub base of the top base (wBase).
wSubBase = Widget_base(wBase, COLUMN=2)

; Left Side with all of the controls
wLeftbase  =  Widget_base(wSubBase,  COLUMN=1,  y_scroll_size=ydim-30,
xsize=190, /scroll, x_scroll_size=191)
;Set  up  two  tabs  -  one  for  plot  controls  and  one  for  indivudual
objects


wTab   =   Widget_tab(wLeftbase,   /ALIGN_TOP,multiline=2,   location=0,
UVALUE='TAB' )
;The First tab
wObjTab= Widget_base(wTab, COLUMN=1, TITLE='Object Control')
;Stuff in the first tab

divider    =    Widget_label(wObjTab,    VALUE='----------------------',
/ALIGN_CENTER)

particleinfo = Widget_label(wObjTab, VALUE='Current Selected Object:',
/ALIGN_CENTER)
;pLabel      = String('Nothing')
labelWid = Widget_label(wObjTab, VALUE='All Objects', /Dynamic_Resize)

;divider    =    WIDGET_LABEL(wObjTab,    VALUE='----------------------',
/ALIGN_CENTER)

rSlider = Widget_slider(wObjTab, MINIMUM=0, MAXIMUM=255, TITLE='Red',
$
```

```
  VALUE=dColor[0],                                    UVALUE='colorSlider',
Event_Pro='DaveColor_Polygons', uName='RSLIDER')
gSlider     =     Widget_slider(wObjTab,     MINIMUM=0,     MAXIMUM=255,
TITLE='Green', $
  VALUE=dColor[1],                                    UVALUE='colorSlider',
Event_Pro='DaveColor_Polygons', uName='GSLIDER')
bSlider = Widget_slider(wObjTab, MINIMUM=0, MAXIMUM=255, TITLE='Blue',
$
  VALUE=dColor[2],                                    UVALUE='colorSlider',
Event_Pro='DaveColor_Polygons', uName='BSLIDER')
tSlider     =     Widget_slider(wObjTab,     MINIMUM=0,     MAXIMUM=255,
TITLE='Transparency', $
  VALUE=255,    UVALUE='alphaSlider',    Event_Pro='DaveColor_Polygons',
UNAME='ASLIDER')
autotrans    =    Widget_button(wObjTab,    Value='Auto    Transparency',
Event_Pro='Polyview_Autotrans', UValue='autotrans', UNAME='AUTOTRANS')

buttonBase = Widget_base(wObjTab, Column=2)
brFront    =    Widget_button(buttonbase,    Value='Bring    to    Front',
Event_Pro='Polyview_Autotrans', UValue='brFront')
brForward    =    Widget_button(buttonbase,    Value='Bring    Forward',
Event_Pro='Polyview_Autotrans', UValue='brForward')
sndBack    =    Widget_button(buttonbase,    Value='Send    to    Back',
Event_Pro='Polyview_Autotrans', UValue='sndBack')
sndBackward   =   Widget_button(buttonbase,   Value='Send   Back   One',
Event_Pro='Polyview_Autotrans', UValue='sndBackward')

;divider = WIDGET_LABEL(wSubLeftBase2, VALUE='----------------------',
/ALIGN_CENTER)
list = ['Fill', 'Wire', 'Points']
fillwire = Cw_bgroup(wObjTab, list, /EXCLUSIVE, Set_Value=0,COLUMN=3
,$
  UValue='fillwire', Event_FUNC='Dave_FillWire'  )
;The second tab
wPlotTab= Widget_base(wTab, COLUMN=1, TITLE='View Control')
;divider   =   WIDGET_LABEL(wLeftBase,   VALUE='----------------------',
/ALIGN_CENTER)
wShowAxis     =     Widget_button(wPlotTab,     Value='Hide     Axis',
Uvalue='AXIS',UNAME='AXISBUT', $
  Event_Pro='Polyview_Properties', /ALIGN_CENTER);, TOOLTIP='Show/Hide
the axis')

divider    =    Widget_label(wPlotTab,    VALUE='----------------------',
/ALIGN_CENTER)
wsubLeftbase1 = Widget_base(wPlotTab, COLUMN=2, /Align_Center)
wTranslate    =    Cw_bgroup(wsubLeftBase1,    ['Rotate','Translate'],
/Exclusive, Set_Value=0,COLUMN=2 ,$
  UValue='Translate', Event_FUNC='Dave_Translate', UNAME='ROT/TRANS' )
wReset = Widget_button(wPlotTab, Value=' Reset ', Uvalue='Reset', $
  Event_Pro='Polyview_Properties',    /ALIGN_CENTER,    TOOLTIP='Reset
Rotation/Translate')


wRescale = Widget_button(wPlotTab, Value='Rescale', Uvalue='Rescale',
$
```

```
  Event_Pro='Dave_ReScale', /ALIGN_CENTER, TOOLTIP='Rescale The Plot')

zoomSlider    =    Widget_slider(wPlotTab,    MINIMUM=0,    MAXIMUM=100,
TITLE='Zoom', $
  VALUE=zoom,   UVALUE='ZSLIDER',   Event_Pro='Polyview_Zoom',   /DRAG,
UNAME='ZOOMSLIDE')

EulerBase = Widget_base(wPlotTab, Row =3)

Getypr, origTransform, ypr

wYaw = Cw_fslider(eulerbase, Value=ypr[0], /EDIT , /DRAG, TITLE='Yaw',
$
  UVALUE = 'Yaw',  Format='(f0.1)', Min=0, Max=360, UNAME="YAW")
wPitch        =    Cw_fslider(eulerbase,    Value=ypr[1],    /EDIT    ,
/DRAG,TITLE='Pitch', $
  UVALUE = 'Yaw',Format='(f0.1)', Min=0, Max=360 , UNAME="PITCH")
wRoll     =     Cw_fslider(eulerbase,     Value=ypr[2],     /EDIT     ,
/DRAG,TITLE='Roll', $
  UVALUE = 'Yaw', Format='(f0.1)', Min=0, Max=360, UNAME="ROLL" )

capBase= Widget_base(wPlotTab, Column=1)
wCaps = Cw_bgroup(capbase, ['Top Cap', 'Bottom Cap'], /NONEXCLUSIVE,
Set_Value=[1,1],COLUMN=2 ,$
  UValue='capswitch', Event_FUNC='Dave_caps', UNAME='CAPBUTTON'  )


;The third.
wProTab=   Widget_base(wTab,   COLUMN=1,   TITLE='Property   Coloring',
sensitive = 1)

DataSelect= Cw_bgroup(wProTab, ['Object Data','Vertex Data', 'Vertex
Color'], /EXCLUSIVE, Set_Value=0,Row=3, $
  UValue='EDITPROP', UNAME='DATATYPE' , /Return_index, /No_Release )
dummy = Widget_label(wProTab, Value = 'Data Column')
clist = Size(object_data)
IF clist[0] EQ 1 THEN clist='0' ELSE clist = Strtrim(Indgen(clist[1]-
1),2)
ColumnSelect       =       Widget_combobox(wProTab,       Value=clist,
UVALUE='EDITPROP', sensitive=1, UNAME='COLUMNSELECT', /Dynamic)

;mx = Max([1.,0.0], Min=mn)
wmxProp   =   Cw_fslider(wProTab,   Value=objdatarange[1],   /EDIT   ,
TITLE='Max Range Property', $
  UVALUE   =   'EDITPROP',      Format='(f0.3)',   Min=objDataRange[0],
Max=objDataRange[1], UNAME='PROPSLIDERMAX', /drag)
wmnProp   =   Cw_fslider(wProTab,   Value=objdatarange[0],   /EDIT   ,
TITLE='Min Range Property', $
  UVALUE   =   'EDITPROP',      Format='(f0.3)',   Min=objDataRange[0],
Max=objDataRange[1],UNAME='PROPSLIDERMIN', /drag)
wPropReset    =    Widget_button(wProTab,    Value='Reset    Sliders',
UNAME='SLIDERRESET', UVALUE='EDITPROP')
wPropReset     =     Widget_button(wProTab,     Value='Reset     Scale',
UNAME='PROPRESET', UVALUE='EDITPROP')
```

```
Loadct, Get_names=ctnames
ctnames[33] = ctnames[33]+' (default)'
dummy = Widget_label(wProTab, Value = 'Color Palette')
wPalette = Widget_combobox(wProTab, Value=ctnames, UVALUE='EDITPROP',
sensitive=1, UNAME='PALETTE')
Widget_control, wPalette, Set_ComboBox_Select=33

dummy = Widget_label(wProTab, Value = 'Display Legends')
list = ['Color Bar','EBSD']
Selectaction=       Cw_bgroup(wProTab,       list,       /NONEXCLUSIVE,
Set_Value=[0,0],Column=2 ,$
  UValue='EDITPROP', UNAME='LEGENDON' , /Return_index )


;Setup the ObjPicker/Selection Picker

divider = Widget_label(wLeftBase, VALUE='----------------------')
Tab  =  Widget_tab(wLeftBase,  multiline=2,  location=0,  UVALUE='TAB',
/align_center)
tab1 = Widget_base(tab, TITLE='Show/Hide')
viewlist = Intarr(N_elements(object_label)+1)+1
list = ['All', Reform( object_label, N_elements(object_label))]
;temp= Widget_base(tab1, align_center=1)
objpicker       =       Cw_bgroup(tab1,       list,       /NONEXCLUSIVE,
Set_Value=viewlist,COLUMN=2 ,$
  /scroll,       y_scroll_size=130,       xsize=150,x_scroll_size=160,
UValue='objhide',   Event_FUNC='Dave_OBJPicker',   UNAME='OBJSHOWHIDE',
/Return_Index  )

tab2 = Widget_base(tab, TITLE='Select', row=2)
;Text = WIDGET_LABEL(tab2, VALUE='Change Object Selection')
selectlist = Intarr(N_elements(object_label)+1)+1
list       =       ['All',       Reform(       object_label[0:nPart-1],
N_elements(object_label[0:nPart-1]))]
;temp1 = Widget_base(tab2, align_center=1)
;temp2 = Widget_base(tab2, align_center=1)
objselector       =       Cw_bgroup(tab2,       list,       /NONEXCLUSIVE,
Set_Value=viewlist[0:nPart],COLUMN=2 ,$
  /scroll,       y_scroll_size=130,       xsize=150,x_scroll_size=160,
UValue='objselect',   Event_FUNC='Dave_OBJPicker',   UNAME='OBJSELECT',
/Return_INDEX  )

list = ['One', 'Add', 'Subtract']
Selectaction=  Cw_bgroup(tab2,  list,  /EXCLUSIVE,  Set_Value=1,Column=3
,$
  UValue='SELECTACTION',                   Event_FUNC='Dave_OBJPicker',
UNAME='SELECTTYPE'  )

; Create the right Base that has the drawing area.
wRightbase = Widget_base(wSubBase)

wDraw    =    Widget_draw(wRightBase,    GRAPHICS_LEVEL=2,    XSIZE=xdim,
YSIZE=ydim, $
  /EXPOSE_EVENTS,  /BUTTON_EVENTS,  UVALUE='DRAW',  RETAIN=0,  RENDERER=0
)
```

```
wGuiBase = Widget_base(wBase, /ROW)
IF N_elements(iansfile) EQ 0 THEN BEGIN
  Widget_control, wBase, /REALIZE, xoffset=10, yoffset=10
  Widget_control, /HOURGLASS

  ; Get the window id of the drawable.
  Widget_control, wDraw, GET_VALUE=oWindow

  ; Save state
  sState = {btndown: 0b,                $     ; Mouse Control
    dragq: 2,               $     ; Drag Quality
    viewPlane:myView,       $     ; View Plane for 10% zoom
    oHolder:oHolder,        $     ; Holder
    oTrack:oTrack,                $     ; Track Ball
    Wdraw:Wdraw,            $     ; Draw
    oWindow:oWindow,        $     ; Window
    oMyPolygons:oPart,            $     ; Polygons
    nPart:nPart,            $     ;  Number  of  particles,  excluding
planes/arrows
    ;objpicker:objpicker,         $     ; Object picker interface
    ;viewList:viewlist,           $     ;   true/flase   list   for   the
current hide value
    objnames:object_label, $     ; Custom objectnames
    origTransform:origTransform, $     ; OG zoom
    oL1:oL1,                $     ; Light Scheme 1
    oL2:oL2,                $     ; Light Scheme 2
    oLOIM:oLOIM,            $     ;OIM Light Scheme
    lightstatus:[0,1],            $     ;Light scheme status
    xAxis:xaxis,            $
    yAxis:yaxis,            $
    zAxis:zaxis,            $
    labelWid:labelWid,            $     ; label of particle
    ;CurrentObj:CurrentObj,       $     ; currently selected object
    dColor:dColor,          $     ; OG Polygon Colors
    uColor:uColorPt,        $     ; Pointer to User Defined Colors
    fillwire:fillwire,      $     ; State of object for points/wire/fill
    aspect:aspect,          $     ; Aspect Ratio
    clipplanes:clipplanes,  $     ;clip planes for top/bottom caps
    wCaps:wcaps,            $     ;widget for caps
    oGroup: oGroup,         $     ; Group - rotating group
    oMicro:oMicro,                $; Microstructure - rotates with group
- scaled
    oNoAxis:oNoAxis,  $; group within oMicro with out axis
    oTop:oTop,              $     ; Topgroup - non rotating
    oView: oView,                 $     ; View
    oViewGroup: oViewGroup,       $     ; View Group
    oOIMLedgend: oOIMLedgend,     $     ; OIMLedgend Group
    oColorLedgend: oColorLedgend,       $     ; OIMLedgend Group
    ;oOrientationText:oOrientationText,     $     ; Some info about the
orientation coloring
    OIM2DVECT: [0,0,1.0],         $     ; OIM 2d color vector
    ;euler:euler,                 $     ; euler rotation angles for
each particle
    oColorbar:oColorBar,          $     ;User property color bar
```

```
    oCAxis:oCAxis,                    $     ;User property color axis
    objDataRange:objDataRange,   $     ;  global  data  range  of  the
object data
    vertDataRange:vertDataRange,$      ;  global  data  range  of  the
vertex data
    oPalette:oPalette}                      ;  object  that  holds  the
palette for the user_property and vert_property coloring



  Widget_control, wBase, SET_UVALUE=sState, /NO_COPY



  Xmanager,    'NRLViewer',    wBase,    Event_Handler='Polyview_Event',
/NO_BLOCK
ENDIF ELSE BEGIN



  Clipboard     =     Obj_new('IDLgrBuffer',     Dimensions=[xdim,ydim],
Graphics_tree=oViewGroup)
  clipboard->Draw, oViewgroup

  clipboard->Getproperty, IMAGE_DATA = snapshot


  Write_bmp, iansfile, snapshot, /RGB
  Heap_free, oHolder
  Heap_free, oOIMLedgend
  Heap_free, oMicro
  ;Ptr_free, uColorPt
  ;Ptr_free, user_prop_pt
  Heap_free, oViewGroup
  Heap_free, Clipboard
  Heap_free, oViewGroup
  Heap_free, oHolder
  Heap_gc
ENDELSE


;Destroy the progress window
Widget_control, startup, /destroy

;PRINT, '  Viewer Launched'


END
```

References

[1] B. W. Reed, B. L. Adams, J. V. Bernier, C. M. Hefferan, A. Henrie, S. F. Li*, et al.*, "Experimental Tests of Stereological Estimates of Grain Boundary Populations," *Acta Materialia,* vol. 60, pp. 2999-3010, 2012.

[2] Special Metals. (2004). INCOLOY Alloy 800H & 800HT. [PDF file]. Available: www.specialmetals.com/documents/Incoloy%20alloys%20800H%20800HT.pdf

[3] ASTM International E112, 2012, "Standard Test Methods for Determining Average Grain Size," ASTM International, West Conshohocken, PA.

[4] Methanex Corporation. (2012). Our Company. [Online]. Available: http://www.methanex.com/ourcompany/index.html

[5] Methanex Corporation. (2008). Methanex Environmental Excellence Report. [PDF file]. Available: http://www.methanex.com/environment/documents/2008_Environmental_Report.pdf

[6] R. Kirchheiner and P. Woelpert, "Niobium in centrifugally cast tubes for petrochemical applications," in *Proceedings of the International Symposium Niobium 2001, May 2, 2001 - December 5, 2001*, Orlando, FL, United states, 2001, pp. 1041-1054.

[7] G. V. Raynor and V. G. Rivlin, "Phase Equilibria in Tron Ternary Alloys," *The Institute of Metals,* vol. 4, 1988.

[8] U.S. Department of Energy, "Next Generation Nuclear Plant Steam Generator and Intermediate Heat Exchanger Materials Research and Development Plan," Idaho National Laboratory, U.S. Department of Energy, 2010

[9] J. Erneman, J. O. Nilsson, H. O. Andren, and D. Tobjork, "Microstructural Evolution During Creep of Alloy 800HT in the Temperature Range 600°C to 1000°C," *Metallurgical and Materials Transactions A,* vol. 40, pp. 539-550, 2009.

[10] M. Venkatraman and J. P. Neumann, "The C-Cr (carbon-chromium) system," *Bulletin of Alloy Phase Diagrams,* vol. 11, pp. 152-159, 1990.

[11] H. Frost and M. Ashby, *Deformation-Mechanism Maps: The Plasticity and Creep of Metals and Ceramics*, 1st ed.: Pergamon Press, 1982.

[12] R. L. Coble, "A Model for Boundary Diffusion Controlled Creep in Polycrystalline Materials," *Journal of Applied Physics,* vol. 34, pp. 1679-1682, 1963.

[13] C. Herring, "Diffusional viscosity of polycrystalline solid," *Journal of Applied Physics,* vol. 21, pp. 437-445, 1950.

[14] J. J. Hoffman and G. Y. Lai, "Metallurgical Evaluation of Alloy 800HT Pigtails," in *Corrosion2005*, 2005.

[15] D. Drabble, "The Effects of Grain Boundary Engineering on the Properties of Incoloy 800H/HT," Ph.D dissertation, Mechanical Engineering, University of Canterbury, New Zealand, 2010.

[16] G. Malakondaiah and P. Rama Rao, "Effect of grain size, grain shape and subgrain size on high temperature creep behaviour," *Defence Science Journal,* vol. 35, pp. 201-217, 1985.

[17] J. G. Y. Chow, P. Soo, and L. Epel, "Creep and fatigue properties of Incoloy 800H in a high-temperature gas-cooled reactor (HTGR) helium environment," presented at the International conference on Alloy 800, Petten, Netherlands, 1978.

[18] E. Hamzah, M. Mudang, and M. Khattak, "Effect of Variation in Grain Size on High Temperature Creep Test of Fe-Ni-Cr Alloy," *Advanced Materials Research,* vol. 845, pp. 51-55, 2014.

[19] T. Yu and H. Shi, "Effects of Grain Size Distribution on the Creep Damage Evolution of Polycrystalline Materials," *Journal of Physics D: Applied Physics,* vol. 43, pp. 1-6, 2010.

[20] B. N. Kim, K. Hiraga, K. Morita, and I. W. Chen, "Rate of Creep due to Grain-Boundary Diffusion in Polycrystalline Solids with Grain-Size Distribution," *Philosophical Magazine,* vol. 85, pp. 2281-2292, 2005.

[21] J. H. Schneibel, R. L. Coble, and R. M. Cannon, "The Role of Grain Size Distribution in Diffusional Creep," *Acta Metallurgica,* vol. 29, pp. 1285-1290, 1981.

[22] S. Onaka, A. Madgwick, and T. Mori, "Kinetics of diffusional creep discussed by energy dissipation and effect of grain-size distribution on the rate equations," *Acta Materialia,* vol. 49, pp. 2161-2168, 2001.

[23] J. Don and S. Majumdar, "Creep cavitation and grain boundary structure in type 304 stainless steel," *Acta Metallurgica,* vol. 34, pp. 961-967, 1986.

[24] D. J. Michel, H. Nahm, and J. Moteff, "Deformation Induced Twin Boundary Crack Formation in Type 304 Stainless Steel," *Materials Science and Engineering,* vol. 11, pp. 97-102, 1973.

[25] C. W. Weaver, "Influence of annealing twins of intergranular creep cracking," *Institute of Metals,* vol. 87, pp. 126-127, 1958.

[26] F. Otto, E. J. Payton, J. Frenzel, and G. Eggeler, "The Effectiveness of Coincidence Site Lattice Criteria in Predicting Creep Cavitation Resistance," *Journal of Materials Science,* vol. 47, pp. 2915-2927, 2012.

[27] C. J. Boehlert, "The Effect of Thermomechanical Processing on the Creep Behavior of Alloy 690," *Materials Science and Engineering: A,* vol. 473, pp. 233-237, 2008.

[28] C. Y. Cui, Y. F. Gu, Y. Yuan, T. Osada, and H. Harada, "Enhanced Mechanical Properties in a New Ni-Co Base Superalloy by Controlling Microstructures," *Materials Science and Engineering: A,* vol. 528, pp. 5465-5469, 2011.

[29] R. D. Doherty, D. A. Hughes, F. J. Humphreys, J. J. Jonas, D. J. Jensen, M. E. Kassner*, et al.*, "Current issues in recrystallization: a review," *Materials Science and Engineering: A,* vol. 238, pp. 219-274, 1997.

[30] M. Avrami, "Kinetics of Phase Change. I General Theory," *The Journal of Chemical Physics,* vol. 7, pp. 1103-1112, 1939.

[31] W. A. Johnson and R. F. Mehl, "Reaction kinetics in processes of nucleation and growth," *American Institute of Mining and Metallurgical Engineers -- Transactions,* vol. 135, pp. 416-442, 1939.

[32] A. K. Sinha, *Physical Metallurgy Handbook*. New York: McGraw Hill, 2003.

[33] J. E. Burke and D. Turnbull, "Recrystallization and grain growth," *Progress in Metal Physics,* vol. 3, pp. 220-292, 1952.

[34] M. P. Anderson, D. J. Srolovitz, G. S. Grest, and P. S. Sahni, "Computer Simulation of Grain Growth - 1. Kinetics," *Acta Metallurgica,* vol. 32, pp. 783-791, 1984.

[35] H. V. Atkinson, "Theories of Normal Grain Growth in Pure Single Phase Systems," *Acta Metallurgica,* vol. 36, pp. 469-491, 1988.

[36] R. T. DeHoff and F. N. Rhines, *Quantitative microscopy*. New York: McGraw-Hill, 1968.

[37] P. Feltham, "Grain growth in metals," *Acta Metallurgica,* vol. 5, pp. 97-105, 1957.

[38] F. N. Rhines and B. R. Patterson, "Effect of the Degree of Prior Cold Work on the Grain Volume Distribution and the Rate of Grain Growth of Recrystallized Aluminum," *Metallurgical Transactions A,* vol. 13, pp. 985-993, 1982/06/01 1982.

[39] A. P. Zhilyaev, J. Gubicza, G. Nurislamova, Á. Révész, S. Suriñach, M. D. Baró*, et al.*, "Microstructural characterization of ultrafine-grained nickel," *physica status solidi (a),* vol. 198, pp. 263-271, 2003.

[40] F. J. Humphreys and M. Hatherly, *Recrystallization and Related Annealing Phenomena*, Second ed. Oxford, UK: Elsevier Ltd, 2004.

[41] M. Hillert, "Kinetics of cementite particle coarsening at 700°C," *Acta Metall,* vol. 13, pp. 227-236, 1965.

[42] N. P. Louat, "On the theory of normal grain growth," *Acta Metallurgica,* vol. 22, pp. 721-724, 1974.

[43] M. Fátima Vaz and M. A. Fortes, "Grain size distribution: The lognormal and the gamma distribution functions," *Scripta Metallurgica,* vol. 22, pp. 35-40, 1988.

[44] M. Groeber, S. Ghosh, M. D. Uchic, and D. M. Dimiduk, "A Framework for Automated Analysis and Simulation of 3D polycrystalline Microstructures. Part 1: Statistical Characterization," *Acta Materialia,* vol. 56, pp. 1257-1273, 2008.

[45] F. J. Humphreys, "Grain and Subgrain Characterisation by Electron Backscatter Diffraction," *Journal of Materials Science,* vol. 36, pp. 3833-3854, 2001.

[46] A. P. Sutton and R. W. Balluffi, *Interfaces in Crystalline Materials*. Oxford, UK: Clarendon Press, 1995.

[47] W. T. Read, *Dislocations in Crystals*. NY: McGraw Hill Book Company, 1953.

[48] W. T. Read and W. Shockley, "Dislocation Models of Crystal Grain Boundaries," *Physical Review,* vol. 78, pp. 275-289, 1950.

[49] D. G. Brandon, "The Structure of High-Angle Grain Boundaries," *Acta Metallurgica,* vol. 14, pp. 1479-1487, 1966.

[50] J. M. Howe, *Interfaces in Materials*. New York: John Wiley & Sons, 1997.

[51] M. Winning and A. D. Rollett, "Transition between low and high angle grain boundaries," *Acta Materialia,* vol. 53, pp. 2901-2907, 2005.

[52] M. L. Kronberg and F. H. Wilson, "Atomic Relationships in the Cubic Twinned State," *Trans. AIME,* vol. 185, p. 501, 1949.

[53] G. Palumbo and K. T. Aust, "Structure-Dependence of Intergranular Corrosion in High Purity Nickel," *Acta Metallurgica,* vol. 38, pp. 2343-2352, 1990.

[54] M. F. Ashby, F. Spaepen, and S. Williams, "The structure of grain boundaries described as a packing of polyhedra," *Acta Metallurgica,* vol. 26, pp. 1647-1663, 1978.

[55] L. E. Murr, R. J. Horylev, and W. N. Lin, "Interfacial Energy and Structure in fcc Metals and Alloys," *Philosophical Magazine,* vol. 22, pp. 515-542, 1970.

[56] A. P. Sutton and R. W. Balluffi, "On Geometric Criteria for Low Interfacial Energy," *Acta Metallurgica,* vol. 35, pp. 2177-2201, 1987.

[57] J. D. Rittner and D. N. Seidman, "<111> Symmetric Tilt Grain Boundary Structures in fcc Metals with Low Stacking-Fault Energies," *Physical Review,* vol. 54, pp. 6999-7015, 1996.

[58] D. Horton, C. B. Thomson, and V. Randle, "Aspects of Twinning and Grain Growth in High Purity and Commercially Pure Nickel," *Materials Science and Engineering: A* vol. A203, pp. 408-414, 1995.

[59] C. B. Thomson and V. Randle, "A Study of Twinning in Nickel," *Scripta Materialia,* vol. 35, pp. 385-390, 1996.

[60] E. M. Lehockey, A. M. Brennenstuhl, and I. Thompson, "On the Relationship Between Grain Boundary Connectivity, Coincident Site Lattice Boundaries, and Intergranular Stress Corrosion Cracking," *Corrosion Science,* vol. 46, pp. 2383-2404, 2004.

[61] D. P. Field, L. T. Bradford, M. M. Nowell, and T. M. Lillo, "The Role of Annealing Twins During Recrystallization of Cu," *Acta Materialia,* vol. 55, pp. 4233-4241, 2007.

[62] B. W. Reed and M. Kumar, "Mathematical Methods for Analyzing Highly-Twinned Grain Boundary Networks," *Scripta Materialia,* vol. 54, pp. 1029-1033, 2006.

[63] U. Wolf, F. Ernst, T. Muschik, M. W. Finnis, and H. F. Fischmeister, "The Influence of Grain Boundary Inclination on the Structure and Energy of Σ3 grain boundaries in copper," *Philosophical Magazine A,* vol. 66, pp. 991-1016, 1992.

[64] M. A. Meyers and C. McCowan, "The Formation of Annealing Twins: Overview and New Thoughts," presented at the Interface Migration and Control of Microstructure, Detroit, 1986.

[65] B. Straumal, S. Polyakov, E. Bischoff, W. Gust, and E. Mittemeijer, "Faceting of Σ3 and Σ9 grain boundaries in copper," *Interface Science,* vol. 9, pp. 287-292, 2001.

[66] R. L. Fullman, "Crystallography and Interfacial Free Energy of Noncoherent Twin Boundaries in Copper," *Journal of Applied Physics,* vol. 22, pp. 456-460, 1951.

[67] S. Dash and N. Brown, "Investigation of Origin and Growth of Annealing Twins," *Acta Metallurgica,* vol. 11, pp. 1067-1075, 1963.

[68] C. M. Sargent, "Twin boundaries in aluminum," *Metallurgical Society of American Institute of Mining, Metallurgical and Petroleum Engineers,* vol. 242, pp. 1188-1190, 1968.

[69] D. Vaughan, "Annealing Twin Interfaces in an Austenitic Stainless Steel," *Philosophical Magazine,* vol. 22, pp. 1003-1011, 1970/11/01 1970.

[70] V. Randle and D. J. Dingley, "Measurement of Boundary Plane Inclination in a Scanning Electron Microscope," *Scripta Metallurgica,* vol. 23, pp. 1565-1570, 1989.

[71] V. Randle and Y. Hu, "The Role of Vicinal Σ3 Boundaries and Σ9 Boundaries in Grain Boundary Engineering," *Journal of Materials Science,* vol. 40, pp. 3243-3246, 2005.

[72] T. Norbygaard and J. B. Bilde-Sorensen, "Σ3 Grain Boundaries in Cu-Ni Subjected to Diffusional Creep," presented at the 9th International Conference on Creep & Fracture of Engineering Materials & Structures, 2001.

[73] J. Bystrzycki, W. Przetakiewicz, and K. J. Kurzydłowski, "Study of annealing twins and island grains in F.C.C. alloy," *Acta Metallurgica et materialia,* vol. 41, pp. 2639-2649, 1993.

[74] M. D. Uchic, M. A. Groeber, D. M. Dimiduk, and J. P. Simmons, "3D Microstructural Characterization of Nickel Superalloys via Serial-Sectioning Using a Dual Beam FIB-SEM," *Scripta Materialia,* vol. 55, pp. 23-28, 2006.

[75] A. C. Lewis, J. F. Bingert, D. J. Rowenhorst, A. Gupta, A. B. Geltmacher, and G. Spanos, "Two- and Three-Dimensional Microstructural Characterization of a Super-Austenitic Stainless Steel," *Materials Science and Engineering: A,* vol. 418, pp. 11-18, 2006.

[76] J. Alkemper and P. W. Voorhees, "Quantitative Serial Sectioning Analysis," *Journal of Microscopy,* vol. 201, pp. 288-394, 2001.

[77] Wright-Patterson Air Force Base, "Digital Representation of Materials Grain Structure," Air Force Research Laboratory, Wright-Patterson Air Force Base, 2010

[78] J. E. Spowart, "Automated serial sectioning for 3-D analysis of microstructures," *Scripta Materialia,* vol. 55, pp. 5-10, 2006.

[79] M. V. Kral and G. Spanos, "Three-dimensional analysis of proeutectoid cementite precipitates," *Acta Materialia,* vol. 47, pp. 711-724, 1999.

[80] A. D. Rollett, S. B. Lee, R. Campman, and G. S. Rohrer, "Three-Dimensional Characterization of Microstructure by Electron Back-Scatter Diffraction," *Annual Review of Materials Research,* vol. 37, pp. 627-658, 2007.

[81] EDAX. (2013). Hikari XP EBSD Camera. [PDF file]. Available: www.edax.com/download/Hikari_XP_PB_LR.pdfCached

[82] V. Randle, "An Investigation of Grain-Boundary Plane Crystallography in Polycrystalline Nickel," *Journal of Materials Science,* vol. 30, pp. 3983-3988, 1995.

[83] V. Randle, M. Caul, J. Fiedler, ouml, and rn, "Twinning and Interfacial Planes in Copper," *Microscopy and Microanalysis,* vol. 3, pp. 224-233, 1997.

[84] V. Randle, P. Davies, and B. Hulm, "Grain-boundary plane reorientation in copper," *Philosophical Magazine A,* vol. 79, pp. 305-316, 1999.

[85] V. Randle, *The Measurement of Grain Boundary Geometry*. London, UK: Institute of Physics Publishing, 1993.

[86] M. Caul, J. Fiedler, and V. Randle, "Grain-boundary plane crystallography and energy in austenitic steel," *Scripta Materialia,* vol. 35, pp. 831-836, 1996.

[87] V. Randle and H. Davies, "A comparison between three-dimensional and two-dimensional grain boundary plane analysis," *Ultramicroscopy,* vol. 90, pp. 153-162, 2002.

[88] H. C. H. Carpenter and S. Tamura, "The Formation of Twinned Metallic Crystals," *The Royal Society of London Proceedings Series A,* vol. 113, pp. 161-182, 1926.

[89] J. E. Burke, "Formation of Annealing Twins," *American Institute of Mining and Metallurgical Engineers,* vol. 188, pp. 1324-1328, 1950.

[90] J. P. Nielsen, "Origin of Annealing Twins," *Acta Metallurgica,* vol. 15, pp. 1083-1085, 1967.

[91] R. L. Fullman and J. C. Fisher, "Formation of Annealing Twins During Grain Growth," *Journal of Applied Physics,* vol. 22, pp. 1350-1355, 1951.

[92] R. Viswanathan and C. L. Bauer, "Formation of Annealing Twins, Faceting, and Grain Boundary Pinning in Copper Bicrystals," *Metallurgical Transactions* vol. 4, pp. 2645-2650, 1973.

[93] H. Gleiter, "The Formation of Annealing Twins," *Acta Metallurgica,* vol. 17, pp. 1421-1428, 1969.

[94] G. Baro and H. Gleiter, "Formation of Annealing Twins," *Zeitschrift fuer Metallkunde,* vol. 63, pp. 661-663, 1972.

[95] J. R. Cahoon, Q. Li, and N. L. Richards, "On the Calculation of Annealing Twin Density," *Scripta Materialia,* vol. 15, pp. 1155-1158, 2006.

[96] C. Pande, M. Imam, and P. Rath, "Study of Annealing Twins in fcc Metals and Alloys," *Metallurgical and Materials Transactions A,* vol. 21, pp. 2891-2896, 1990.

[97] R. A. Varin and J. Kruszynska, "Control of Annealing Twins in Type 316 Austenitic Stainless Steel," *Acta Metallurgica,* vol. 35, pp. 1767-1774, 1987.

[98] M. A. Meyers and L. E. Murr, "A Model for the Formation of Annealing Twins in F.C.C. Metals and Alloys," *Acta Metallurgica,* vol. 26, pp. 951-1962, 1978.

[99] P. J. Goodhew, "Annealing Twin Formation by Boundary Dissociation," *Metal Science,* vol. 13, pp. 108-112, 1979.

[100] S. Y. Lee, Y. B. Chun, J. W. Han, and S. K. Hwang, "Effect of Thermomechanical Processing on Grain Boundary Characteristics in Two-Phase Brass," *Materials Science and Engineering: A,* vol. 363, pp. 307-315, 2003.

[101] W. Wang and H. Guo, "Effects of Thermo-Mechanical Iterations on the Grain Boundary Character Distribution of Pb–Ca–Sn–Al Alloy," *Materials Science and Engineering: A,* vol. 445–446, pp. 155-162, 2007.

[102] C. R. M. Grovenor, D. A. Smith, and M. J. Goringe, "Nucleation and migration of high angle grain boundaries in bilayer foils II: Migration," *Thin Solid Films,* vol. 74, pp. 269-279, 1980.

[103] A. R. Jones, "Annealing Twinning and the Nucleation of Recrystallization at Grain Boundaries," *Journal of Materials Science,* vol. 16, pp. 1374-1380, 1981.

[104] P. Gerber, "EBSD Study of Annealing Twinning during Recrystallization of Cold Rolled Copper," *Materials Science Forum,* vol. 495, pp. 1303-1308, 2005.

[105] T. Baudin, A. L. Etter, and R. Penelle, "Annealing Twin Formation and Recrystallization Study of Cold-Drawn Copper Wires from EBSD Measurements," *Materials Characterization,* vol. 58, pp. 947-952, 2007.

[106] B. R. Kumar, S. K. Das, B. Mahato, A. Das, and S. Ghosh Chowdhury, "Effect of Large Strains on Grain Boundary Character Distribution in AISI 304L Austenitic Stainless Steel," *Materials Science and Engineering: A,* vol. 454–455, pp. 239-244, 2007.

[107] T. Watanabe, "Approach to Grain Boundary Design for Strong and Ductile Polycrystals," *International Journal of Structural Mechanics and Materials Science,* vol. 11, pp. 47-84, 1984.

[108] V. Randle, "Twinning-Related Grain Boundary Engineering," *Acta Materialia,* vol. 52, pp. 4067-4081, 2004.

[109] M. Kumar, A. J. Schwartz, and W. E. King, "Microstructural Evolution During Grain Boundary Engineering of Low to Medium Stacking Fault Energy fcc Materials," *Acta Materialia,* vol. 50, pp. 2599-2612, 2002.

[110] S.-L. Lee and N. L. Richards, "The Effect of Single-Step Low Strain and Annealing of Nickel on Grain Boundary Character," *Materials Science and Engineering: A,* vol. 390, pp. 81-87, 2005.

[111] Q. Li, J. R. Cahoon, and N. L. Richards, "Effects of Thermo-Mechanical Processing Parameters on the Special Boundary Configurations of Commercially Pure Nickel," *Materials Science and Engineering: A,* vol. 527, pp. 263-271, 2009.

[112] C. Thomson and V. Randle, "The Effects of Strain Annealing on Grain Boundaries and Secure Triple Junctions in Nickel 200," *Journal of Materials Science,* vol. 32, pp. 1909-1914, 1997.

[113] Q. Li, B. M. Guyot, and N. L. Richards, "Effect of Processing Parameters on Grain Boundary Modifications to Alloy Inconel 718," *Materials Science and Engineering: A,* vol. 458, pp. 58-66, 2007.

[114] V. Randle, M. Coleman, and G. Owen, "Evolution of the Grain Boundary Network as a Consequence of Deformation and Annealing," in *International Symposium on Fundamentals of Deformation and Annealing, September 5, 2006 - September 7, 2006*, Manchester, United kingdom, 2007, pp. 35-44.

[115] D. J. Drabble, C. M. Bishop, and M. V. Kral, "A Microstructural Study of Grain Boundary Engineered Alloy 800H," *Metallurgical and Materials Transactions,* vol. 42, pp. 763-772, 2011.

[116] V. Randle, "Mechanism of Twinning-Induced Grain Boundary Engineering in Low Stacking-Fault Energy Materials," *Acta Materialia,* vol. 47, pp. 4187-4196, 1999.

[117] B. W. Reed, R. W. Minich, R. E. Rudd, and M. Kumar, "The Structure of the Cubic Coincident Site Lattice Rotation Group," *Acta Crystallographica Section A: Foundations of Crystallography,* vol. A60, pp. 263-277, 2004.

[118] M. Kumar, C. A. Schuh, and W. E. King, "Connectivity in random grain boundary networks," in *Electron Microscopy: Its Role in Materials Science, March 2, 2003 - March 6, 2003*, San Diego, CA, United states, 2003, pp. 51-58.

[119] M. Kumar, W. E. King, and A. J. Schwartz, "Modifications to the microstructural topology in f.c.c. materials through thermomechanical processing," *Acta Materialia,* vol. 48, pp. 2081-2091, 2000.

[120] G. V. Voort, *Metallography, Principles and Practice*. New York: McGraw-Hill, 1984.

[121] ASTM International E2627, 2010, "Standard Practice for Determining Average Grain Size Using Electron backscatter Diffraction (EBSD) in Fully Recrystallized Polyscrystalline Materials," ASTM International, West Conshihocken, PA.

[122] A. J. Schwartz, M. Kumar, B. L. Adams, and D. P. Field, *Electron backscatter diffraction in materials science*: Springer, 2009.

[123] M. H. Alvi, S. Cheong, H. Weiland, and A. D. Rollett, "Microstructural evolution during recrystallization in hot rolled Aluminum Alloy 1050," in *Proc. 1st Intl. Symp. on Metallurgical Modeling for Aluminum Alloys, TMS, Pittsburgh*, 2003, pp. 191-197.

[124] Y. Estrin, M. Heilmaier, and G. Drew, "Creep Properties of an Oxide Dispersion Strengthened Nickel- Base Alloy. The Effect of Grain Orientation and Grain Aspect Ratio," *Materials Science and Engineering. A,* vol. 272, pp. 163-173, 1999.

[125] S. G. Chowdhury and R. Singh, "The influence of recrystallized structure on the snesitization behaviour of a stable austenitix stainless steel (AISI 316L)," *Scripta Materialia,* vol. 58, pp. 1102-1105, 2008.

[126] P. J. Wilbrandt, "Recrystallization texture in terms of multiple twinning," *Physica status solidi (a),* pp. 411-418, 1980.

[127] B. Alexandreanu and G. S. Was, "A Priori Determination of the Sampling Size for Grain-Boundary character Distribution and Grain-Boundary Degradation Analysis," *Philosophical Magazine,* vol. 81, pp. 1951-1965, 2001.

[128] S. I. Wright and R. J. Larsen, "Extracting Twins from Orientation Imaging Microscopy Scan Data," *Journal of Microscopy,* vol. 205, pp. 245-252, 2002.

[129] J. H. Cho, A. D. Rollett, and K. H. Oh, "Determination of a Mean Orientation in Electron Backscatter Diffraction Measurements," *Metallurgical and Materials Transactions A,* vol. 36A, pp. 3427-3438, 2005.

[130] V. Randle, "A Methodology for the Grain Boundary Plane Assessment by Single-Section Trace Analysis," *Scripta Materialia,* vol. 44, pp. 2789-2794, 2001.

[131] V. Randle, "The Influence of Annealing Twinning on Microstructure Evolution," *Journal of Materials Science,* vol. 40, pp. 853-859, 2005.

[132] R. G. Morse, "The effect of heat-treatment upon the physical properties and the microstructure of medium carbon steel," *Transactions of the American Institute of Mining Engineers,* pp. 729-751, 1899.

[133] ASTM International E930, 1999, "Standard Test Methods for Estimating the Largest Grain Observed in a Metallographic Section (ALA Grain Size) " ASTM International, West Conshohocken, PA.

[134] ASTM International E1181 2002, "Standard Test Methods for Characterizing Duplex Grain Sizes Grain Size) " ASTM International, West Conshohocken, PA.

[135] ASTM International E1382 1997, "Standard Test Methods for Determining Average Grain Size Using Semiautomatic and Automatic Image Analysis " ASTM International, West Conshohocken, PA.

[136] S. I. Wright, "A parametric study of electron backscatter diffraction based grain size measurements," *Practical Metallography,* vol. 47, pp. 16-33, 2010.

[137] G. Bockstiegel, "A Simple Formula for the Calculation of Spatial Size Distributions from Data Found by Lineal Analysis," in *Stereology*, H. Elias, Ed., ed: Springer Berlin Heidelberg, 1967, pp. 193-194.

[138] R. G. Cortés, A. O. Sepúlveda, and W. O. Busch, "On the determination of grain-size distributions from intercept distributions," *Journal of Materials Science,* vol. 20, pp. 2997-3002, 1985.

[139] S. Saltikov, "The Determination of the Size Distribution of Particles in an Opaque Material from a Measurement of the Size Distribution of Their Sections," in *Stereology*, H. Elias, Ed., ed: Springer Berlin Heidelberg, 1967, pp. 163-173.

[140] H. D. Lewis, K. L. Walters, and K. A. Johnson, "Particle size distribution by area analysis: modifications and extensions of the Saltykov method," *Metallography,* vol. 6, pp. 93-101, 1973.

[141] Y. Takayama, N. Furushiro, T. Tozawa, H. Kato, and S. Hori, "A significant method for estimation of the grain size of polycrystalline materials," *Materials Transactions JIM,* vol. 32, pp. 214-221, 1991.

[142] K. Matsuura and Y. Itoh, "Estimation of three-dimensional grain size distribution in polycrystalline material," *Materials Transactions JIM,* vol. 32, pp. 1042-1047, 1991.

[143] J. C. Tucker, L. H. Chan, G. S. Rohrer, M. A. Groeber, and A. D. Rollett, "Comparison of grain size distributions in a Ni-based superalloy in three and two dimensions using the Saltykov method," *Scripta Materialia,* vol. 66, pp. 554-557, 2012.

[144] V. Randle, "Application of EBSD to the Analysis of Interface Planes: Evolution Over the Last Two Decades," *Journal of Microscopy,* vol. 230, pp. 406-143, 2008.

[145] L. Chan, "Synthetic Three-Dimensional Voxel-Based Microstructures that Contain Annealing Twins," Ph.D dissertation, Material Science and Engineering, Carnegie Mellon University, Pittsburgh, PA, 2010.

[146] C. S. Smith, "Grains, phases, and interphases: an interpretation of microstructure," *Trans. Metall. Soc. AIME,* vol. 175, pp. 15-51, 1948.

[147] H. Conrad, M. Swintowski, and S. L. Mannan, "Effect of cold work on recrystallization behavior and grain size distribution in titanium," *Metallurgical transactions A,* vol. 16 A, pp. 703-708, 1985.

[148] J. W. Christian, *Theory of Transformations in Metal and Alloys, Part I*, 3rd ed. Oxford, U.K.: Pergamon Press, 2003.

[149] A. R. Paul, M. C. Naik, and K. N. G. Kaimal, "Mass transport of iron in Incoloy 800," *Materials Science and Technology,* vol. 7, pp. 8-11, 1991.

[150] K. T. Aust and J. W. Rutter, "Annealing twins and coincidence site boundaries in high-purity lead," *Transactions of the American Institute of Mining and Metallurgical Engineers,* vol. 218, pp. 1023-1028, 1960.

[151] P. J. Wilbrandt, "On the role of annealing twin formation in the recrystallisation texture development," *Scripta Metallurgica et Materialia,* vol. 27, pp. 1485-1492, 1992.

[152] D. C. Crawford and G. S. Was, "The Role of Grain Boundary Misorientation in Intergranular Cracking of Ni-16Cr-9Fe in 360 °C Argon and High-Purity Water," *Metallurgical Transactions,* vol. 23, pp. 1195-1206, 1992.

[153] W. L. Grube and S. R. Rouze, "The Origin, Growth and Annihilation of Annealing Twins in Austenite," *Canadian Metallurgical Quarterly,* vol. 2, pp. 31-52, 1963.

[154] C. Minkwitz, C. Herzig, E. Rabkin, and W. Gust, "Inclination dependence of gold tracer diffusion along a $\Sigma 3$ twin grain boundary in copper," *Acta Materialia,* vol. 47, pp. 1231-1239, 1999.

[155] ASTM Internations E8, 2013, "Standard Test Methods for Tension Testing of Metallic Materials," ASTM Internations, West Conshohocken, PA.

[156] K. Buchanan, "The Effects of Long-Term Isothermal Ageing on the Microstructure of HP-Nb and Hp-NbTi Alloys," Ph.D dissertation, Mechanical Engineering, University of Canterbury, New Zealand, 2013.

[157] T. Sato, "Power-Law Creep Behaviour in Magnesium and its Alloys," Ph.D dissertation, Mechanical Engineering, University of Canterbury, New Zealand, 2008.

[158] (October 2013). LVDT Basics. Available: http://www.macrosensors.com/lvdt_tutorial.html

[159] B. Alexandreanu, B. Spencer, V. Thaveeprungsriporn, and G. Was, "The Effect of grain boundary character distribution on the high temperature deformation behaviour of Ni-16Cr-9Fe alloys," *Acta Materialia,* vol. 51, pp. 3831-3848, 2003.

[160] E. Arzt, M. Ashby, and R. Verrall, "Interface Controlled Diffusional Creep," *Acta Matall,* vol. 31, pp. 1977-1989, 1983.

[161] T. Norbygaard, "Studies of Grain Boundaries in Materials Subjected of Diffusional Creep," Ph.D dissertation, Riso National Laboratory, Copenhagen University, Copenhagen, Denmark, 2002.

[162] B. Burton and G. L. Reynolds, "Defense of diffusional creep," *Materials Science and Engineering A,* vol. A191, pp. 135-141, 1995.

[163] Y. Tanaka, A. Sato, and T. Mori, "Stress assisted nucleation of α ”precipitates in Fe・ N single crystals," *Acta Metallurgica,* vol. 26, pp. 529-540, 1978.

[164] E. M. Lehockey and G. Palumbo, "On the creep behaviour of grain boundary engineered nickel," *Materials Science and Engineering A,* vol. A237, pp. 168-172, 1997.

[165] C. B. Carter and S. M. Holmes, "Stacking Fault Energy of Nickel," *Philosophical Magazine,* vol. 35, pp. 1161-1172, 1977.

[166] L. Vitos, J. O. Nilsson, and B. Johansson, "Alloying effects on the stacking fault energy in austenitic stainless steels from first-principles theory," *Acta Materialia,* vol. 54, pp. 3821-3826, 2006.

[167] T. Sato and M. Kral, "Electron Backscatter Diffraction Mapping of Microstructural Evolution of Pure Magnesium During Creep " *Metall. Mat. Trans. A,* vol. 29, 2008.

[168] D. M. Saylor, J. Fridy, B. S. El-Dasher, K.-Y. Jung, and A. D. Rollett, "Statistically representative three-dimensional microstructures based on orthogonal observation sections," *Metallurgical and Materials Transactions A,* vol. 35, pp. 1969-1979, 2004.

[169] A. Brahme, M. Alvi, D. Saylor, J. Fridy, and A. Rollett, "3D reconstruction of microstructure in a commercial purity aluminum," *Scripta Materialia,* vol. 55, pp. 75-80, 2006.

[170] A. C. Lewis, K. A. Jordan, and A. B. Geltmacher, "Determination of Critical Microstructural Features in an Austenitic Stainless Steel Using Image-Based Finite Element Modeling," *Metallurgical and Materials Transactions A,* vol. 39, pp. 156-164, 2008.

[171] J. C. Tucker, L. H. Chan, G. S. Rohrer, M. A. Groeber, and A. D. Rollett, "Tail Departure of Log-Normal Grain Size Distributions in Synthetic Three-Dimensional Microstructures," *Metallurgical and Materials Transactions,* vol. 43, pp. 2810-2822, 2012.

[172] M. E. Abd El-Azim, "Correlation between tensile and creep data in Alloy 800H at 850C," *Journal of Nuclear Materials,* vol. 231, pp. 146-150, 1996.

[173] M. C. Naik, A. R. Paul, K. N. G. Kaimal, and K. S. Venkateswarlu, "Mass transport of cobalt and nickel in Incoloy-800," *Journal of Materials Science,* vol. 25, pp. 1640-1644, 1990.

[174] A. R. Paul, K. N. G. Kaimal, M. C. Naik, and S. R. Dharwadkar, "Lattice and grain boundary diffusion of chromium in superalloy Incoloy-800," *Journal of Nuclear Materials,* vol. 217, pp. 75-81, 1994.

[175] B. Raeisinia and C. W. Sinclair, "A representative grain size for the mechanical response of polycrystals," *Materials Science and Engineering A,* vol. 525, pp. 78-82, 2009.